(2)

"AN INTERACTIVE, OPTIMIZATION BASED, COMPUTER GRAPHICS SOFTWARE PACKAGE - PHASE II"

CONTRACT (F08635-88-C-0063)

FINAL TECHNICAL REPORT AND USERS MANUAL

for the period October 1987 to October 1989

MARK D. LANDON, JERRY L. UDY

APTEK, INC.
1257 Lake Plaza Drive
Colorado Springs, CO 80906

October 25, 1989

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

June 1990 Final Oct 87 - Oct 89

An Interactive, Optimization-Based, Computer Graphics Software Package-Phase II

PE: 65502F
C: F08635-88-C-0063

Mark Landon, Jerry Udy
MSD Program Manager: G. Allen Baker (MSD/XRC)

APTEK, Inc.
1257 Lake Plaza Drive
Colorado Springs, CO 80906

Deputy for Development Plans
Munitions Systems Division (MSD/XRC)
Eglin AFB, FL 32542-5000

MSD-TR-90-10

This report was not edited nor published by AFATL/DOIR.

Approved for public release; distribution is unlimited.

This report documents the work that was completed to develop the software tools and geometric data base that will help Air Force Design Engineers take aircraft and missiles from concept to test specimen. Navgraph is a geometric modeling program that allows the user to define complex, convex and non-convex geometric solid objects for use in the various modules developed. A summary of the capabilities are given here:

NAVGRAPH can be used to create geometric models of aircraft, weapons,submunitions, pylons, racks, etc. to whatever detail is necessary for the problem being studied.

NAVGRAPH can be used to recall any geometric model(s) from the existing data base (via DBMERGE) to define the desired complex model to whatever detail is necessary. The existing data base consists of the models developed under this contract, mainly, an F-15 aircraft, fuel tank, racks, rails, bombs, missiles, and submuntions.

CALIPER can be used to study physical fit compatibility. For example, CALIPER can calculate the separation distance between a missile fin and an aircraft wing or detect and calculate the interference distance and direction of internal components with respect to the interior of a missile body. The information derived here can be used to translate the interfering object and remove the interference. (KR)

Physical Fit, Compatibility, Optimization, Packaging, Interference, Solid Modeling, Computer Graphics CAD/CAM

135

UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

13. ABSTRACT (CONCLUDED)

MASSPROP will calculate the mass properties (including surface area, volume, mass, center of gravity and moments of inertia) of a NAVGRAPH model.

PACKER (along with OPTDES) can be used to package submunitions and/or internal components in a weapon with constraints on certain mass properties. For example, find the optimal placement and orientation of internal components and/or submunitions that satisfy constraints on the roll, pitch and yaw moments of inertia as well as constraints on the location of the center of gravity with respect to the center of pressure.

NAVGRAPH can be used to generate finite elements (polygonal facets) on geometric models for use in aerodynamic, radar cross section, stress, etc. prediction software codes.

NAVGRAPH can accept finite element data from other Air Force data base formats such as ASE and VUMADM.

NAVGRAPH can be used to generate computer color graphic displays (line drawings or smooth rendering) of all the above upon demand including interference highlighting.

In addition, formating software can be used to translate (via IGES) the data base to and from formats compatible with other software codes for generating computer controlled milling machine instructions.

Table of Contents

Report Overview

This report discusses the work completed under Air Force SBIR contract F08635-88-C-0063 titled, "An Interactive, Optimization-Based, Computer Graphics Software Package - Phase II" by APTEK, Inc. Part 1 contains a technical description of the work done. Part 2 contains the users manual and tutorial with examples.

The overall objective for the Phase II work was to develop and deliver the software tools that will help the Air Force aircraft and missile designer take a weapon from concept to test specimen quickly and accurately and to provide an initial data base of important existing designs.

The software package consists of the following modules described here. Figure 1 shows the inter-relationships of the modules.

- NAVGRAPH is the geometric modeling module that allows the user to define points, lines, surfaces and solids that are used to define the solid objects that make up a physical fit compatibility study or optimal packaging problem. NAVGRAPH also has, among other things, a friendly user interface and finite element pre- and post-processing capabilities. Examples of the modeling capability are contained throughout this report.
- CALIPER is the separation and interference calculator. It accesses a previously defined NAVGRAPH data base (that contains a model developed for a physical fit compatibility study) and then calculates the needed separation and interference distances between user defined objects. CALIPER can be run in NAVGRAPH so one does not have to exit a NAVGRAPH session to detect and display interferences.
- PACKER is the optimal packaging software. It 1) accesses a previously defined model in a NAVGRAPH data base, 2) calculates the initial separations and interferences between all objects and all objects and container walls, 3) computes the composite mass properties for the whole model, 4) allows the user to "setup" the desired packaging problem by defining variables and functions for the optimization, and 5) allows the user to run OPTDES to find the optimal packaging solution for the defined problem.
- OPTDES, a commercial optimization software package, is the optimization software used to solve the packaging problem. It has several robust and efficient mathematical optimization search routines that help find the values of variables defining optimal design solutions.
- MASSPROP calculates the mass properties of NAVGRAPH models.
- DBMERGE allows the user to merge models from several NAVGRAPH data base files together into one file for a physical fit compatibility study or optimal packaging problem.

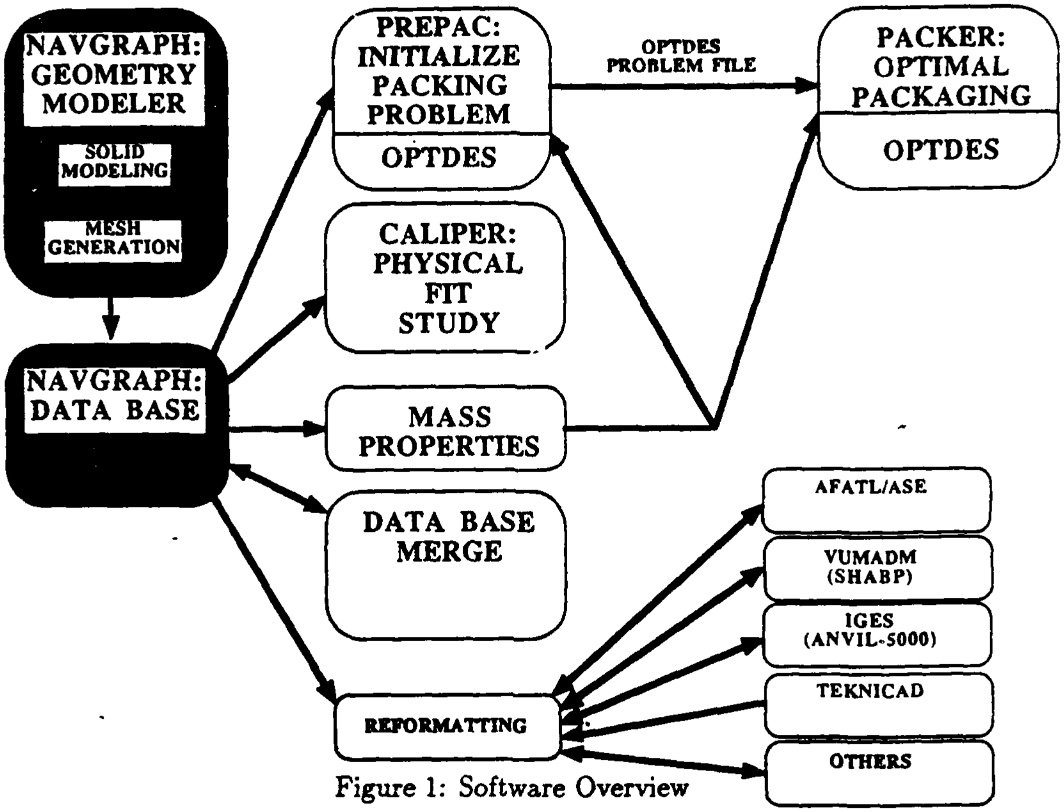


Figure 1: Software Overview

The following capabilities are available to the user of the software package outlined above:

1. NAVGRAPH can be used to create geometric models of aircraft, weapons, submunitions, pylons, racks, etc. to what ever detail is necessary for the problem being studied.

2. NAVGRAPH can be used to recall any geometric model(s) from the existing data base (via DBMERGE) to define the desired complex model to what ever detail is necessary. The existing data base consists of the models developed under this contract, namely, an F-15 aircraft, fuel tank, racks, rails, bombs, missiles, and submunitions. One can use the existing data base and DBMERGE to build complex models by assembling simpler models without re-entering their data by adding new subassemblies, or by modifying old data.

3. CALIPER can be used to study physical fit compatibility. For example, CALIPER can calculate the separation distance between a missile fin and an aircraft wing or detect and calculate the interference distance and direction of internal components with respect to the interior of a missile body. The information derived here can be used to translate the interfering object and remove the interference.

4. MASSPROP calculates the mass properties (including surface area, volume, mass,

center of gravity and moments of inertia) of a NAVGRAPH model. These functions can be used as design functions to match or limit user defined design criteria.

5. PACKER (along with OPTDES) can be used to package submunitions and/or internal components in a dispenser missile cargo bay with constraints on certain mass properties. For example, it will find the optimal placement and orientation, of internal components and/or submunitions, that satisfies constraints on the roll, pitch and yaw moments of inertia as well as constraints on the location of the center of gravity with respect to the center of pressure.

6. NAVGRAPH can be used to generate finite elements (polygonal facets) on geometric models for use in several prediction software codes such as aerodynamic, radar cross section, stress, etc..

7. NAVGRAPH can provide finite element and geometric data for other Air Force data base formats such as AFATL/ASE and VUMADM. NAVGRAPH can also provide geometric model data (via IGES) to ANVIL for numerically controlled milling machine instructions. NAVGRAPH can accept finite element and geometry data from other Air Force data bases for further model development or graphic display.

8. NAVGRAPH can be used to generate computer color graphic displays (line drawings or smooth rendering) of all the above upon demand including interference highlighting.

Part 1

The Technical Description

The Technical Description

The Phase II work consisted of the following main areas, namely: 1) Graphics, Mass Properties and Hardware Compatibility, 2) Interference Calculation and Optimal Packaging, 3) Translator Software Development, and 4) Data Base Generation, 5) Other Capabilities and Enhancements. The results accomplished in each area are discussed below.

The discussion contained herein assumes an understanding of solid geometric modeling as provided by NAVGRAPH and an understanding of the optimization techniques as provided by OPTDES. A complete discussion of these two packages are contained in their respective users manuals that accompany this report or can be obtained from Brigham Young University or APTEK.

Graphics, Mass Properties and Hardware Compatibility

Hardware compatibility.

The complete software package was developed to be compatible to run on a MicroVax computer (and all DEC-VMS). All graphics output is compatible with Tektronix 4010, 4109, and 4129 graphics terminals. The Tektronix 4129 terminal driver allows the use of the local hardware (rotate, zoom, pan, etc.) commands. NAVGRAPH can accept data input with the Tektronix 4957 and 4958 digitizers as well as with the use of thumbwheels and traditional keyboard input. Tektronix 4692 and 4693 copiers can be used to get hardcopies of any display created by NAVGRAPH.

All of the software can also run under operating systems other than DEC-VMS, such as the UNIX operating system. Different graphics devices, other than Tektronix, simply need a different graphics driver module containing the graphics machine dependent calls. There are many drivers available, and new ones are easily developed.

Mass properties calculator.

The capability was developed to calculate the mass properties (volume, mass, center of gravity, and moments of inertia) of any NAVGRAPH solid model.

A NAVGRAPH session is run to create the desired solids and to group them to define the object being modeled. An object is defined as a group of solids that will rotate and translate together. In preparation for mass properties calculation, each solid in the object is meshed with linear (i.e. no midside nodes) solid finite elements. Different linear solid finite elements (hexahedrons, pentahedrons, wedges, and tetrahedrons) are generated automatically depending on the degree of degeneracy of the solid being meshed. If there are no degeneracies, all elements will be hexahedrons. If there is one surface degenerating to a line, there will be a row of wedge elements along the degenerate surface. If a surface degenerates to a point then there will be a layer of pentahedrons. MASSPROP was written to handle any of these cases.

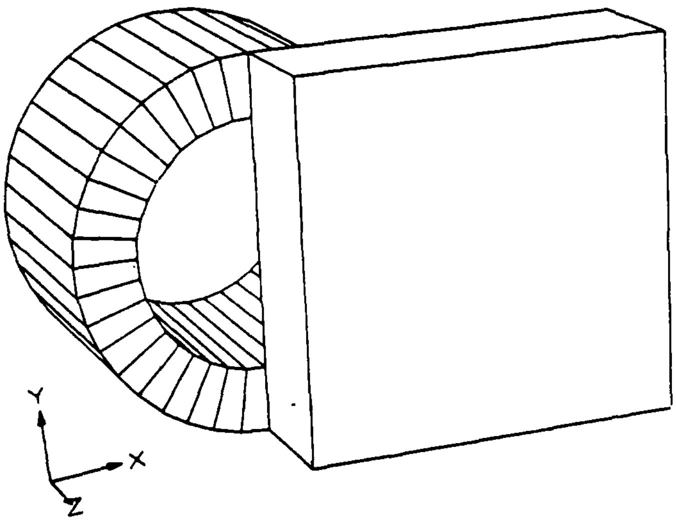
Once the object has all of its solids meshed with solid finite elements, MASSPROP calculates the mass properties element by element. MASSPROP uses the Gauss Quadrature method of integration to evaluate the volume (v_i), mass (m_i) and centroid ($\bar{x}_i, \bar{y}_i, \bar{z}_i$), of each *ith* element. Once this information is known, the composite mass properties of the whole object are simply found with the proper summation formula for each mass property. The following equations show how this summation is performed with respect to the global coordinate system.

$$
\begin{aligned}
v &= \sum_i v_i & (1)\\
m &= \sum_i m_i & (2)\\
\bar{x} &= \frac{\sum_i m_i \bar{x}_i}{m} & (3)\\
\bar{y} &= \frac{\sum_i m_i \bar{z}_i}{m} & (4)\\
\bar{z} &= \frac{\sum_i m_i \bar{y}_i}{m} & (5)\\
I_{xx} &= \sum_i m_i(\bar{y}_i^2 + \bar{z}_i^2) & (6)\\
I_{yy} &= \sum_i m_i(\bar{x}_i^2 + \bar{z}_i^2) & (7)\\
I_{zz} &= \sum_i m_i(\bar{x}_i^2 + \bar{y}_i^2) & (8)\\
I_{xy} &= -\sum_i m_i(\bar{x}_i \bar{y}_i) & (9)\\
I_{xz} &= -\sum_i m_i(\bar{x}_i \bar{z}_i) & (10)\\
I_{yz} &= -\sum_i m_i(\bar{y}_i \bar{z}_i) & (11)
\end{aligned}
$$

The values of the moments of inertia with respect to the local coordinate system with an origin at $(\bar{x}, \bar{y}, \bar{z})$ are simply found with the parallel axis theorem,

$$
\begin{aligned}
I_{x'x'} &= I_{xx} - m(\bar{y}^2 + \bar{z}^2) & (12)\\
I_{y'y'} &= I_{yy} - m(\bar{x}^2 + \bar{z}^2) & (13)\\
I_{z'z'} &= I_{zz} - m(\bar{x}^2 + \bar{y}^2) & (14)\\
I_{x'y'} &= I_{xy} + m(\bar{x}\bar{y}) & (15)\\
I_{x'z'} &= I_{xz} + m(\bar{x}\bar{z}) & (16)\\
I_{z'z'} &= I_{yz} + m(\bar{y}\bar{z}) & (17)
\end{aligned}
$$

Figure 2 shows a an object that was modeled, meshed and had its mass properties



	MASSPROP:	CLOSED FORM	% ERROR
SPECIFIC WEIGHT:	490.0000		
GRAVITY CONST. :	32.20000		
DENSITY :	15.21739		
TOTAL VOLUME :	16.82173	16.85398	0.2
TOTAL MASS :	255.9828	256.4736	0.2
GLOBAL XBAR :	0.8025337	.800997	0.2
GLOBAL YBAR :	1.6473875E-07	0.0	-
GLOBAL ZBAR :	-0.1974664	-.1990023	0.8
GLOBAL IXX :	403.3832	404.8	0.3
GLOBAL IYY :	711.5353	714.0	0.3
GLOBAL IZZ :	706.2109	708.7	0.3
GLOBAL IXY :	1.1503696E-05	0.0	-
GLOBAL IXZ :	-102.7173	-103.1	0.3
GLOBAL IYZ :	4.1633844E-05	0.0	0.0
CENTROID IXX :	393.4017	394.8	0.3
CENTROID IYY :	536.6854	538.3	0.3
CENTROID IZZ :	541.3425	542.6	0.3
CENTROID IXY :	4.5346773E-05	0.0	-
CENTROID IXZ :	-143.2838	-143.7	0.3
CENTROID IYZ :	3.3306631E-05	0.0	-

Figure 2: The mass properties of a model calculated with MASSPROP and compared with the closed form solution.

computed by MASSPROP. The mass properties from MASSPROP are compared to those obtained by closed form solution. The values calculated by MASSPROP are exact for models with straight sides, and are approximate for models with curved sides. As more solid elements are used along a curved side the solution approaches the exact solution. As a general rule, 10 element subdivisions along a curved side will result in less than one percent deviation from the exact solution. More complex models are solved by simply creating more solids, meshing the solids into finite elements, and including the solids into the desired object.

Once MASSPROP has calculated the mass properties for an object they are printed out to the screen and written to an output file on disk.

The mass properties are then available to PACKER for use as optional constraints and objective functions in the optimal packaging problem, as discussed later in the optimal packaging with mass properties section.

Highlight interferences.

The NAVGRAPH code was enhanced to graphically highlight the interferences between objects. This was accomplished by adding two commands to the NAVGRAPH global group (GLOB GROU) menu. While in the global group menu, one needs to type "CALI" for CALIPER to detect and calculate the location and magnitude of all interferences. Once CALIPER is finished with its calculations, the user types "HIGH" and defines the desired color of the highlight, then a flashing vector is drawn that shows the location, direction and magnitude of all interferences. If the user translates, rotates, or scales any group in the data base, CALIPER would then need to be rerun for HIGHLIGHTs to be displayed again.

Geometric data base methodologies.

The ability to use NAVGRAPH to create solid models of specific shapes, such as ellipsoids and tangent ogive bodies of revolution, was studied. The methodologies used to create such models are documented.

The solid entity is a volume bounded by, at most, six surface entities or twelve edge lines. A surface entity in NAVGRAPH is a bi-linear blended Coon's patch from its four edge lines, thus the midspan of a curved surface may not represent the desired surface exactly. The degree to which a NAVGRAPH surface represents a real surface depends on the ability of the bi-linear interpolation to give points on the NAVGRAPH surface that do not vary from where the points would be on the real surface. To get a good representation of a real surface with parametric surfaces, one may need to provide more information by breaking up the surface into more surface segments and providing more lines in the definition. This gives the model more information and less distance between lines in which to interpolate the surface. One must keep this in mind as he creates surfaces from lines and solids from surfaces and lines, especially for bodies of revolution. The trade-off here is more calculations for increased accuracy. That is, more separation/interference calculations are performed where more solids are needed to make complex solid shapes.

An example is given here of the modeling of a solid ellipsoid. Let us model an ellipsoid that is 5.0 inches long along the global x axis, 3.0 inch tall along the global y axis, and 7.0 inches wide along the global z axis. The lines defining a surface segment of the solid ellipsoid are created with the conic line option. This option requires three points (two end points and third point defining the plane) and a parameter defining its relative altitude (0.5 gives a true parabola, .41375 gives a very close ellipse approximation). The steps are listed here.

1. Define 9 points in order at (5,0,0), (5,3,0), (0,3,0), (0,3,7), (0,0,7), (5,0,7). (5,0,0), (5,0,0) and (5,0,0)
2. Define line 1 as a conic with points 1,2,3
3. Define line 2 as a conic with points 3,4,5
4. Define line 3 as a conic with points 5,6,1
5. Define line 4 by mirroring line 1 about zx plane.

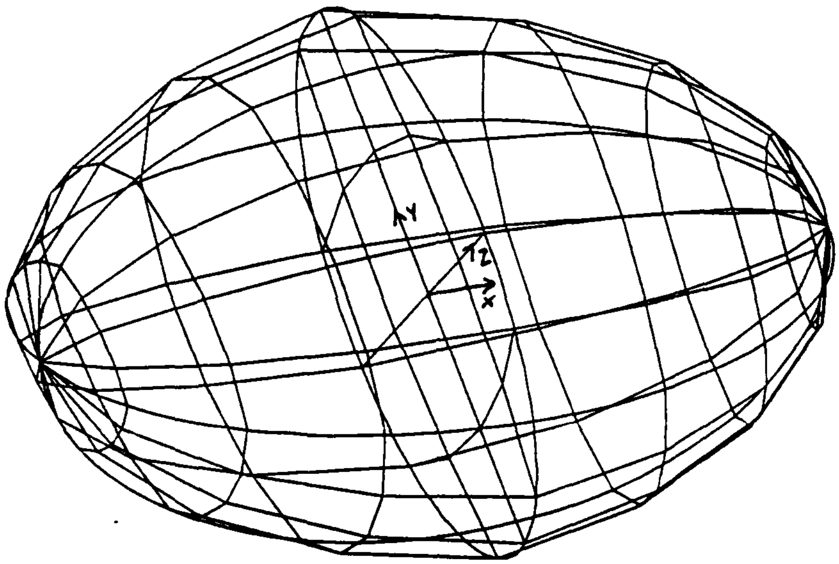


Figure 3: A parametric solid representation of an Ellipsoid.

6. Define line 5 by mirroring line 3 about xy plane.
7. Define line 6 by mirroring line 2 about xy plane.
8. Define line 7 by mirroring line 2 about zx plane.
9. Define line 8 by mirroring line 6 about zx plane.
10. Define lines 9,10,11,12 as a degenerate straight lines between point 1 and point 7, point 7 and point 8, point 8 and point 9, point 9 and point 1.
11. Define solid 1 from lines 6,8,7,2 on the front face, lines 9,10,11,12 on the back face and lines 5,4,3,1 connecting.
12. Define solid 2 by mirroring solid 1 about the yz plane.
13. Put solids 1 and 2 into a group and use the GROUP ROTATE and GROUP TRANSLATE commands to put ellipsoid in the desired orientation (see Figure 3)

Notice that the ellipsoid was modeled with 2 solids. This is a good example of using more solids to represent the shape more exactly. The steps needed to create a tangent ogive body of revolution and other shapes are contained in the tutorial.

The general methodologies studied and learned as a result of the work done in the data base creation, CALIPER examples and PACKER examples have been documented and supplied in the users manual/tutorial. Basically, however, the following items should be kept in mind.

1. Look ahead and decide how the desired model can be broken up into piecewise solids each bound by 12 edges.
2. Using data from drawing, notions, etc., create the points that will make the lines (edges) necessary to make the solids.
3. Be familiar with all of NAVGRAPH's command such as scale, rotate, sweep, translate, etc..
4. Don't be afraid to 'jump in' and learn by using NAVGRAPH.
5. If something does not work the way it seems that it should, try a 'work around' method to arrive at the same results.

CALIPER: Separation and Interference Distances for Physical Fit Study.

A fundamental need exists for a robust capability to calculate the minimum separation distance between two parametrically defined solid objects and, if interference is detected (e.g. the minimum distance is less than or equal to zero), calculate the interference distance between the two solid objects. This need has expressed itself in many different disciplines, such as robotics path planing, animation, and now the physical fit compatibility and optimal packaging problems. The problem is similar in all cases, and has received some attention in the recent past.

Most of the work has been with polyhedral or polygonal geometry where the minimum distance is determined with an exhaustive search of distances between the nodes of one object and the nodes of another, and between the nodes of one object and the polygonal sides of another object, etc.. This leads to a massive amount of data and computations for each pair of objects, especially if arbitrarily shaped, curved sided objects are used. Accuracy is limited in this case since polyhedrons are flat sided approximations to the shape being modeled. The closer the objects get to each other the larger the percent error in the distance calculation. With discrete nodal points and flat sided facets, a smooth function for interference and separation is not possible.

Other work has been done where the shapes are mapped into pixels and if a pixel is occupied by more that one object, collision is detected. The measure of interference is done by counting pixels of overlap and a scalar value of area for the 2-D problem is supplied. This method has several limitations, 1) it requires a large amount of storage for each object to be rasterized into pixels, 2) it has no vector data for direction of the interference, and 3) it has no smooth function of interference and separation distance.

Another area that has received a lot of attention is the area of computing the intersections of two parametrically defined solids, surfaces and/or lines. This area of study is exciting and provides important information for solid geometric modeling, but the result of such methods does not give the information needed for a physical fit study or an optimal packaging problem. For example, where two solids intersect a geometric volume or surface entity is produced, where two surfaces intersect a line is produced, and where two lines intersect points are produced. The resulting entities do not give the necessary measure of interference or separation distances that are needed for studying physical fit compatibility and performing optimal packaging problems.

All methods discussed heretofore have similar limitations when applied to physical fit compatibility and optimal packaging problems, that is they do not give a smooth function containing the direction and magnitude of the separation and/or interference between two solid objects.

A new and innovative method to calculate separation and interference distances between parametrically defined complex solids is presented here. The algorithms have been programmed and incorporated into the software module named CALIPER.

The Solid Modeling Convention.

A brief summary of the solid modeling convention used is presented here.

The point entity is the basic geometric entity and is defined by its three global coordinates (x, y, z).

A line is defined by its cubic parametric coefficients and can represent straight lines, quadratic parametric, cubic parametric, and spline segments. A line can also be an arc segment and a complex line created by merging simple lines. The Cartesian coordinates of any point on a cubic parametric line are given by the cubic equations:

$$x(s) = \sum_{i=0}^{3} a_i s^i \tag{18}$$

$$y(s) = \sum_{i=0}^{3} b_i s^i \tag{19}$$

$$z(s) = \sum_{i=0}^{3} c_i s^i \tag{20}$$

where:

x, y, z	=	Cartesian coordinates of a point.
s	=	parametric coordinate between 0 and 1 along the line. (could also be parameters t or u).
$a's, b's, c's$	=	parametric coefficients.

Cartesian coordinates on arcs are calculated directly from the closed form expression for an arc. For complex lines, each line segment is mapped into the total complex line according to its length so that the complex line also has parametric limits between 0 and 1. With this mapping it is possible to calculate the (x, y, z) coordinates for any parameter value s (between 0 and 1) along any line definition.

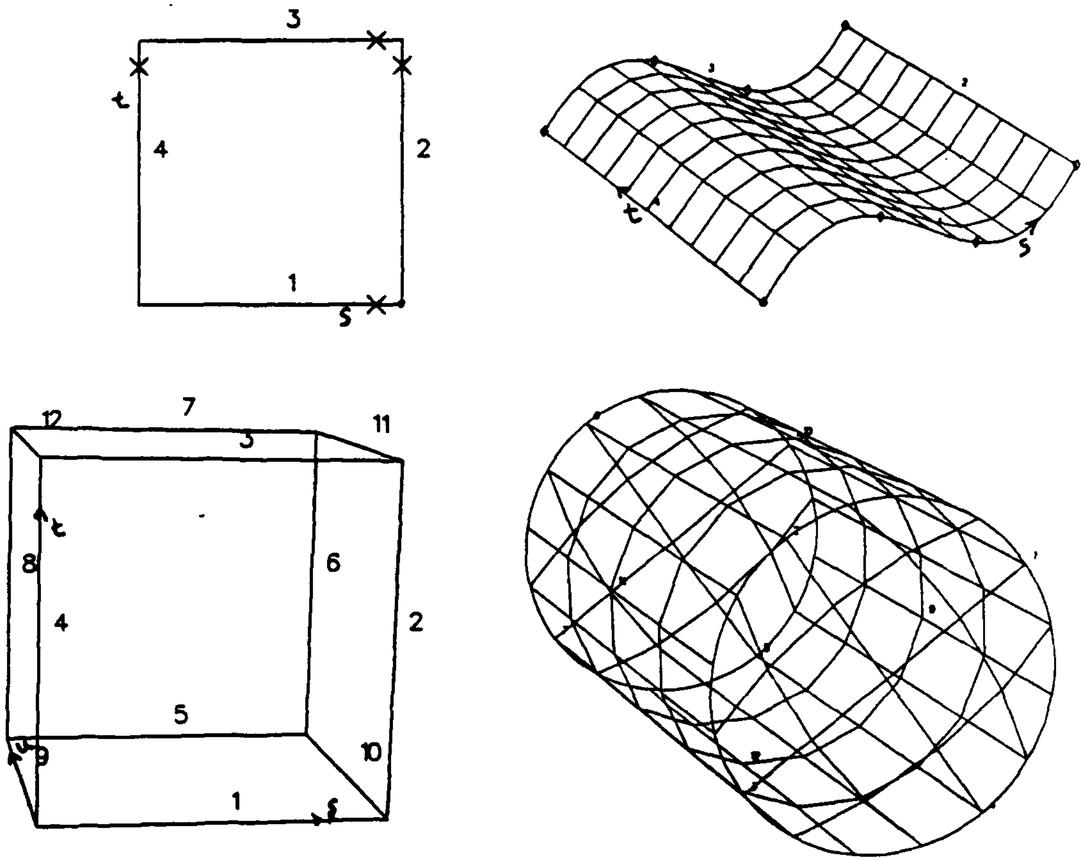


Figure 4: A surface is mapped parametrically into a unit square, and a solid into a unit cube.

A surface entity is defined parametrically with a bilinear blended Coon's patch between four lines connected in the proper order and direction. The lines forming the boundary of the surface can be any of the types above including the complex type. The four lines are mapped into a unit square with parameters s and t (see Figure 4). A point on the surface is calculated with the Coon's interpolation formula. Eight points are needed for the formula (four midside points and four surface corner points).

$$\begin{aligned} x(s,t) &= (1-t)f_1 + sf_2 + (1-s)f_3 + sf_4 \\ &\quad -x(0,0)(1-s)(1-t) - x(1,0)s(1-t) - x(0,1)(1-s)t - x(1,1)st \end{aligned} \tag{21}$$

(and similar equations for $y(s,t)$ and $z(s,t)$) where:

$$\begin{aligned} f_i &= x \text{ on line } i \ (i = 1 \text{ to } 4) \text{ as calculated in (1).} \\ x(0,0) &= x \text{ at the } s = 0 \text{ and } t = 0 \text{ corner.} \\ x(1,0) &= x \text{ at the } s = 1 \text{ and } t = 0 \text{ corner.} \\ x(0,1) &= x \text{ at the } s = 0 \text{ and } t = 1 \text{ corner.} \\ x(1,1) &= x \text{ at the } s = 1 \text{ and } t = 1 \text{ corner.} \end{aligned}$$

A solid entity is defined parametrically in (s,t,u) by bounding the volume it represents with 6 surfaces or with 12 line edges mapped into a unit cube (see Figure 4). Calculation of any Cartesian coordinate within the solids parametric limits is done in a similar fashion to that of surfaces only with a tri-linear blended Coon's patch interpolation between 20 points (eight corners points and 12 midside points). The equation is given here:

$$\begin{aligned} x(s,t,u) &= (1-t)(1-u)f_1 + (1-t)uf_2 + tuf_3 + t(1-u)f_4 + (1-s)(1-u)f_5 \\ &\quad +(1-s)uf_6 + suf_7 + s(1-u)f_8 + (1-s)(1-t)f_9 + (1-s)tf_{10} \\ &\quad +stf_{11} + s(1-t)f_{12} + C(s,t,u) \end{aligned} \tag{22}$$

where:

$$\begin{aligned} C(s,t,u) &= -2\{(1-s)(1-t)(1-u)x(0,0,0) + (1-s)(1-t)ux(0,0,1) + \\ &\quad (1-s)t(1-u)x(0,1,0) + (1-s)tux(0,1,1) + s(1-t)(1-u)x(1,0,0) \\ &\quad s(1-t)ux(1,0,1) + st(1-u)x(1,1,0) + stux(1,1,1) \end{aligned}$$

(and similar equations for $y(s,t,u)$ and $z(s,t,u)$) and where:

$$\begin{aligned} f_i &= x \text{ on line } i\ (i = 1 \text{ to } 12) \text{ as calculated in (1).} \\ x(0,0,0) &= x \text{ at the } s=0,\ t=0 \text{ and } u=0 \text{ corner.} \\ x(1,0,0) &= x \text{ at the } s=1,\ t=0 \text{ and } u=0 \text{ corner.} \\ x(0,1,0) &= x \text{ at the } s=0,\ t=1 \text{ and } u=0 \text{ corner.} \\ x(0,0,1) &= x \text{ at the } s=0,\ t=0 \text{ and } u=1 \text{ corner.} \\ x(1,1,0) &= x \text{ at the } s=1,\ t=1 \text{ and } u=0 \text{ corner..} \\ x(1,0,1) &= x \text{ at the } s=1,\ t=0 \text{ and } u=1 \text{ corner.} \\ x(0,1,1) &= x \text{ at the } s=0,\ t=1 \text{ and } u=1 \text{ corner.} \\ x(1,1,1) &= x \text{ at the } s=1,\ t=1 \text{ and } u=1 \text{ corner.} \end{aligned}$$

This representation of solid geometry is commonly referred to as the a boundary representation of geometric solids. Note that the interpolation is a smooth continuous function in most cases defined with any simple line types and at least continuous in all cases defined with any complex line types. The continuous property is important for the numerical method used in calculating the minimum separation and maximum interference distances.

Finally, an object is made by grouping parametrically defined solids together so they rotate and translate as a single entity. This is commonly referred to as a boolean add in Constructive Solid Geometry (CSG). This ability significantly increases the ability to model very complex, non-convex, arbitrarily shaped objects.

Separation Distance Calculation and Interference Detection.

In the separation distance calculation, two objects are defined by grouping one or more parametrically defined solids together for each object then the minimum distance between them is calculated. The given information is the geometric information as defined above (i.e. the mapping between parametric space (s, t, u) and Cartesian space (x, y, z) for each solid). The following optimization problem is solved for each solid in the first object with respect to each solid in the second object.

Find:

$$s_1, t_1, u_1, s_2, t_2, u_2$$

that minimizes:

$$D = ((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2)^{\frac{1}{2}}$$

such that:

$$\begin{aligned} 0 \leq s_1 \leq 1 \\ 0 \leq t_1 \leq 1 \\ 0 \leq u_1 \leq 1 \\ 0 \leq s_2 \leq 1 \\ 0 \leq t_2 \leq 1 \\ 0 \leq u_2 \leq 1 \end{aligned}$$

where:

s_1, t_1, u_1	= parametric coordinates of solid 1
s_2, t_2, u_2	= parametric coordinates of solid 2
x_1, y_1, z_1	= Cartesian coordinates of solid 1 from s_1, t_1, u_1
x_2, y_2, z_2	= Cartesian coordinates of solid 2 from s_2, t_2, u_2
D	= the distance between the two Cartesian points.

The optimization problem is simply the search for the two points that define the minimum distance between them. The constraints force the parametric coordinates to remain in their respective solids. If an object has more that one solid then this process is repeated for all combinations of solids in object 1 with respect to all solids in object 2. The minimum of all separation distances for all combinations is kept as the minimum separation between the two objects. This definition is the same as that used in the preceding Phase I SBIR study except for the grouping of solids into objects. The use of many solids per objects significantly increases the ability to model complex objects and still get separation distances. If the minimum distance between any pair of solids is computed to be zero or less, then interference has been detected. The calculation of the interference distance is then carried out as discussed in the next section.

The generalized reduced gradient method was employed to solve the distance minimization problem. This solution has been found to be very robust and accurate. There have been literally thousands of test cases run with no failures.

Interference Distance Calculation

Once the separation distance has been detected to be less than or equal to zero, the interference can be calculated. A new definition of interference is presented here.

Given two objects that are interfering, interference is defined as the distance needed to translate the second object away from the first object along some vector until the two objects just touch (i.e. incipient separation). This definition implies both direction and magnitude. The direction of interference may be defined arbitrarily by the user or by default as the direction between the centroids of the two objects.

The user defined direction of interference provides the direction and the magnitude of the translation necessary to remove the interference between two objects. The default direction between the centroids of the objects is best used in the optimal packaging problem.

The interference calculation is also defined as an optimization problem. For any pair of solids that are detected to interfere in the separation calculation, the following problem is solved:

Find:

$$s_1, t_1, u_1, s_2, t_2, u_2, T$$

that maximizes:

$$T = \text{translation of object 2 with respect to object 1}$$

such that:

$$\begin{aligned}
&0 \le s_1 \le 1 \\
&0 \le t_1 \le 1 \\
&0 \le u_1 \le 1 \\
&0 \le s_2 \le 1 \\
&0 \le t_2 \le 1 \\
&0 \le u_2 \le 1 \\
&x_1 = x_2 \\
&y_1 = y_2 \\
&z_1 = z_2
\end{aligned}$$

Where:

s_1, t_1, u_1 = parametric coordinates of solid 1
s_2, t_2, u_2 = parametric coordinates of solid 2
x_1, y_1, z_1 = Cartesian coordinates of solid 1 from s_1, t_1, u_1
x_2, y_2, z_2 = Cartesian coordinates of solid 2 from s_2, t_2, u_2

Note that the additional equality constraints insure that the second solid cannot translate away from the first solid any further than that amount needed to reach incipient separation. The distance T is the magnitude of the translation as well as the negative magnitude of interference. Therefore, interference is always ≤ 0.

This definition of interference is an important improvement over the definition that was presented in the Phase I work. It gives the same interference whether solid 1 is translated with respect to solid 2 or vice versa. It also handles the case where there is full penetration of one object into another (even when identical objects are spatially coincident). It allows a continuous function from separation through incipient separation to interference.

There were some interesting alternative approaches studied before the above definition was decided upon. One method involved shrinking the two objects with a scaling transformation until incipient separation was reached. This method failed in handling the full penetration situation. Another method translated the second object far away from the first object (along the same translation vector as described above) then iteratively translated the object back along the translation vector (with the repeated separation distance calculations) until the objects just touched. This method gave the same results as the chosen method but required much more cpu time.

The generalized reduced gradient method was also employed to solve the translation distance maximization problem. This solution has also been shown to be very robust and accurate. Here again, many test cases have been run with no failures.

CALIPER Examples

An example is included here to show how the calculation of separation and interference distances with CALIPER can help the concept missile designer solve physical fit compatibility problems. The use of CALIPER to help optimally place objects into containers is discussed in the next section.

It is desired to find the orientation of a conceptual bomb on an F-15 wing/pylon-/rack assembly such that there are no interferences during stowage, take-off or landing. It is not desirable to built a proto-type just for compatibility study. First the bomb is modeled with NAVGRAPH. Next, DBMERGE is used to combine the NAVGRAPH data base files containing models of the F-15 wing with 'swept' gear, the runway, the pylons, the rack, and the conceptual bomb (simulated here with an MK84) into one NAVGRAPH file (see Figure 5). A NAVGRAPH session is run and the CALIPER

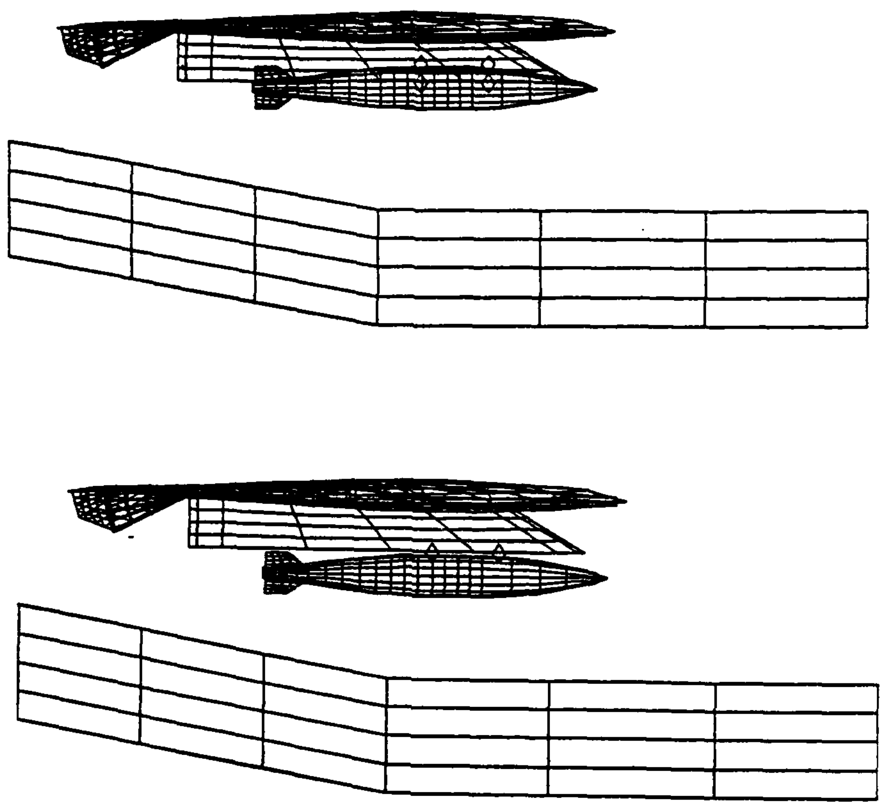


Figure 5: CALIPER calculates the separation and interference distances between weapon and wing/pylon/landing strip assembly.

command issued. Giving the two object names (the bomb and the wing/pylon/rack assembly in this case) and the direction of translation for the interference calculation, CALIPER returns the distance needed to move the bomb such that there is no more interference. Figure 5 shows the distance and location of the interference (a flashing vector is actually shown on the color monitor). The user simply translates the bomb along the direction defined and with the magnitude calculated to remove the interference. Then CALIPER can be run again to make sure there are no other interferences caused by the bomb's recent translation. The process is repeated for each weapon and piece of equipment to be stowed on the aircraft.

This process can also be thought of as representing the volume available for stores on an aircraft by showing the 'volume of exception' available to the stores placement design engineer.

The accuracy obtained by CALIPER in predicting the separation and interference distances is within $\approx$ .001 units for the worst case observed, however, most cases are much better. This is well within the limits set by the Air Force for this development, that is, within $\frac{1}{8}$ inch for submunition to submunition distances and $\frac{1}{4}$ inch for stores to aircraft distances.

PACKER with rotations and translation.

A fundamental need exists for the ability to automatically package objects into containers. The optimal packaging problem arises constantly in our lives. Anyone who has ever packed a trunk or suitcase understands the difficulty of placing as many objects as possible into a limited amount of space. Packaging is generally performed by trial and error which is time consuming and inefficient. In industry packaging problems are becoming increasingly important as companies seek to minimize their costs and increase efficiency. These problems are diverse and range from part layout in the textiles and sheet-metal industries to packaging freight in the transportation industry. The Air Force sees this problem arise when packaging internal components and submunitions into missile cargo bays. Consequently, automation of the packaging problem has received a great deal of interest. The latest related work in this field is reviewed here.

Two-dimensional problems have received considerable attention from researchers. These generally involve the placement of planar shaped objects inside some bounding area with the objective of minimizing the distances between objects. For instance in the textile industry this would apply to cutting material so as to waste as little as possible Another application is the design of integrated circuits where individual components are arranged to produce the smallest design possible. In general, two approaches have been used in solving the optimal packaging: numerical optimization and heuristic algorithms.

Most all work done in this area has been done in the two-dimensional layout problems where the objects packaged are represented with polygons which can have an arbitrary number of edges including non-convex shapes. The design variables are the x and y translations and the one rotation of each polygon. The objective was usually to minimize the area needed to bound the polygons without overlap.

Very little was found in the literature on work done in the three-dimensional realm of optimal packaging. One project has addressed whether a given shape would fit in a given size box or and to find the minimum size box that would fit around a given object. The work was limited to one object in a container.

The Phase I work preceding this Phase II work showed the feasibility of using optimization techniques and solid geometric modeling software to package 3-D convex shapes into 3-D containers. Even though feasibility was shown, some limitations were experienced. One fundamental problem was with the use of Euler angles for rotational degrees of freedom. It was found that the optimization routines were often unable to rotate the objects and the process would stop prematurely. Another problem was that an object could only be represented with one arbitrary solid which limited the freedom to model very complex shapes.

The Phase II work has done much to improve the robustness and functionality of PACKER while at the same time keeping the necessary computing resources at a reasonable limit. The state of the software is detailed here by describing how the many tasks completed during Phase II have been incorporated into the software program

PACKER. We first start by outlining the properties of PACKER and then discuss the steps taken to optimally package objects into container with spatial relationships. The next main section will add the use of mass properties.

Geometric modeling convention

The method of representing 3-D arbitrarily shaped solid objects is the same as the geometry modeling convention as used for CALIPER described in the previous main section. The ability to group arbitrarily shaped solids together to form objects of very complex, non-convex, arbitrary shapes is one of the most important improvements over the Phase I work. Essentially, the user now can model any shape needed and is only limited by his own imagination and ability to use the modeling software.

Rotational degrees of freedom and the quaternion

The question of how to rotate objects properly in the optimization process is an important one. This next section discusses the work done on the general 3-D packaging problem addressing the rotational degrees of freedom.

A fundamental component of the solution to the packaging problem is the ability to change the position and orientation of the objects being packaged. It was proven by Euler that the position of a rigid body may be described by a translation and one rotation about an arbitrary axis. However, as shown in Appendix I, Euler derived the rotation formula in terms of three independent rotations. These rotations know as Euler angles are unsuitable for use in numerical optimization.

Euler angles suffer from singularities. The most severe is known as "gimbal lock", where in certain orientations, a rotational degree of freedom is lost. Euler angles suffer from gimbal lock because they ignore the interdependence of the axes of rotation. When applying a series of translations to an object the final position of the object may be described by one translation. Clearly a series of translations along one axis of the coordinate system are totally independent of any translation along any of the other two axes. Rotations are not like translations; translations add while rotations are multiplied and the axes of a series of rotations are not independent. For example, the x axis is not a linear combination of the y and z axis. The fact that rotation axes multiply with a cross product is called the confounding of axes. Euler angles ignore this confounding of axes when trying to achieve an orientation by rotation about three independent axes. Ignoring this cross-product interdependence of rotation axes causes gimbal lock by aligning two of the three axes.

Singularities are not unique to Euler angles. It has been shown that any three independent parameters can not describe orientation and be both global and nonsingular. A minimum of four variables are needed to uniformly describe orientation and be nonsingular. So normally when using Euler angles a general rotation matrix is used which has the singularity at an orientation which hopefully is seldom needed.

These problems with Euler angles make them poorly suited for use with numerical optimization. Another approach to describing rotations is the use of quaternion calculus. Quaternions have been around for about 150 years and were virtually forgotten until they came to be used in the spacecraft dynamics and in robotics for trajectory planning.

Quaternions solve the problems associated with Euler angles in representing orientations. Quaternions can achieve any orientation as a single rotation around an arbitrary axis so that rotations are independent for any orientation. The quaternion describes the object's new orientation from its original position. If another change in orientation is desired from the intermediate position, a quaternion may again be applied to the object with the final quaternion being given by the product of the first and second quaternions. Quaternions are able to do this because the cross product is preserved in the quaternion product just as they are in rotations.

The optimal packaging problem

The packaging problem is posed as an optimization problem in the form:

Find:	rotations and translations for each object
Minimize or Maximize:	Some user defined objective function
Subject to:	Constraints that no object interfere with each other and that no object interferes with container walls.

The design variables in the packaging problem are the three rotational and the three translational degrees of freedom per object. All design variables are applied to the object with respect to its original position. Objects are rotated via the quaternion about the quaternion's direction vector passing through the objects parametric center. This in effect rotates the object θ degrees, about an axis through the object's local centroid.

Quaternions were applied to the packaging problem by setting the components of the axis vector (n_1, n_2, n_3) as design variables. A quaternion has four degrees of freedom (θ, n_1, n_2, n_3) , so in order to eliminate one degree of freedom, the angle of rotation θ was defined as the magnitude of the axis vector in radians. The rotation angle is therefore restricted to always being a positive quantity, however, this is not a problem. With four variables, a quaternion is able to represent all possible orientations twice, since a positive angle and a positive axis vector (θ, n_1, n_2, n_3) are equivalent to a negative angle and a negative vector $(-\theta, -n_1, -n_2, -n_3)$. By constraining the angle to remain positive the quaternion is still able to represent any possible orientation. The other design variables are the x, y, z translations for each object.

The constraint functions are the distances between objects and the distances between objects and container walls, which may be either a separation or an interference. If the objects interfere, the distance is negative, and if the objects are separated, the distance is positive. Therefore, in the packaging problem all distances between objects and between objects and container walls are constrained to be positive so that there are no interferences.

The objective function, in many cases, is to minimize or maximize the separation distances between objects and a container wall, or the distances between the object centroids and a container wall. The separation distance is defined as described in CALIPER. If several objects are to be packaged to the bottom of a container the objective function would be computed as the sum of the separation distances of the objects with the bottom container surface of the container. The optimization procedure then minimizes the sum of the separation distances by translating and rotating the objects.

The following steps are taken to solve a packaging problem.

Step 1: Create and initializing a starting design.

The user creates the initial design in NAVGRAPH by modeling the objects and the container, and then by translating and rotating the objects in NAVGRAPH into their intended initial position. In fact, one might try to solve the packaging problem without the aid of the optimization routines simply by trial and error within NAVGRAPH. This can be accomplished by moving the objects until they appear to be in an optimal position. The CALIPER command in the NAVGRAPH GROUP menu is the interference detector. If one did not have the interference detection capability, he would need to try to detect the interference visually by viewing the model from various positions. This is virtually impossible for a model of any real size with a large number of objects.

The most efficient solution to the packaging problem is to have the user and PACKER complement each other. The major advantages of PACKER is that it guarantees that interferences do not occur: an extremely difficult task for humans. One disadvantage of numerical optimization is that the algorithms may get trapped in local minima and terminate the search before the global minimum is reached, whereas a human can provide heuristics and avoid most local minimum. A simple example of how optimization algorithms can get caught in local minima is shown in Figure 6. The objective function of this problem was to minimize the distance between the centroid of the rectangle and the bottom of the container. The constraints are that the rectangle is not allowed to penetrate the sides of the container. It is obvious that the rectangle may be rotated 90 degrees and translated down to the bottom of the container but the optimization routines are unable to find this design because the objective function would initially increase or constraint functions would be violated with any rotation or translation.

Even a simple packaging problem may contain many local minima. Therefore, it is apparent that the packaging routines become much more efficient if they are given a good starting design. The better the starting design, the better the final design and the quicker the optimal design is found. It is also apparent that if the optimization problem is guided by expertise and common sense, many of the local minima will be avoided. This means that even though this tool is powerful and useful, if the user can couple

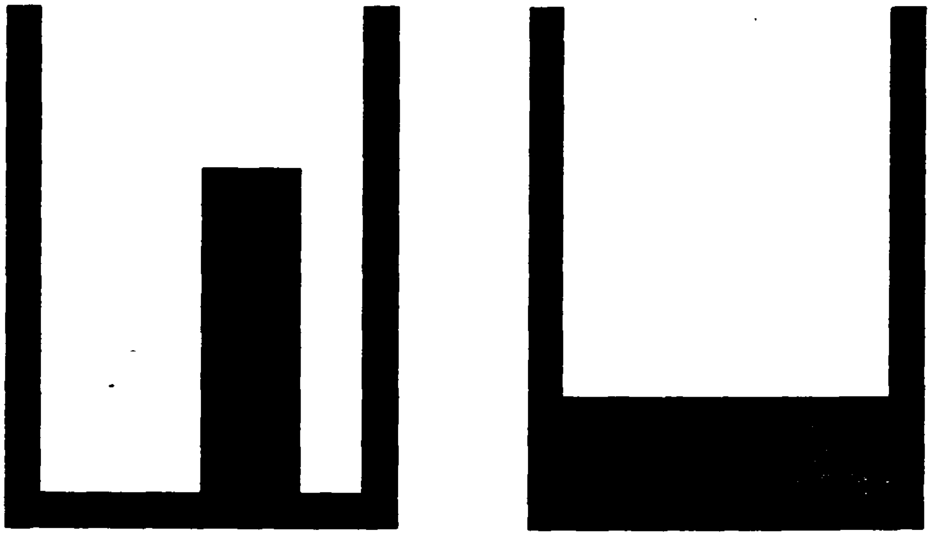


Figure 6: Example of local optima in a packaging problem.

his expertise, common sense, and creativity with the program, he may solve even more complex problems than either program or user could solve separately.

Once the model has been created, it needs to be initialized by running a version of CALIPER called PREPAC. PREPAC simply calculates all separation distances between all objects and between all objects and container walls, then writes an OPTDES problem file containing all initial analysis variables and analysis functions for input into SETUP and eventually PACKER.

Step 2: The Setup.

To setup the packaging problem, one executes the SETUP module of OPTDES that reads in the problem file from PREPAC then defines design variables, constraint functions, and the objective functions for the desired optimization problem.

The design variables are defined as a subset of the analysis variables. If design variables are not defined, the object to which the analysis variables belong will not move in the direction of the design variables omitted. The user can use this to his advantage in problem setup to keep objects from moving with respect to certain degrees of freedom (e.g. let objects translate but not rotate, etc.) Upper and lower bounds are placed on the variables. These bounds are important as they greatly effect the scaling of the problem. By varying the bounds on the rotations and translations it is possible to favor one over the other. This becomes important in highly constrained problems where there are tight fits between objects and container wall.

Constraint functions are defined as inequality functions by restricting the functions to be either greater, or less than some value. In the packaging problem, the separation distances were constrained to be greater than some very small positive value. This is because it was found that by putting a small buffer zone around the objects, (of say, less that .01 inches) less interference calculations were needed near the optimal solution. This practice saves execution time.

The objective function is chosen, and then minimized or maximized depending on the desired optimization problem. More than one function may be mapped to one objective function so the objective function may be defined as the sum of all the separation distances between all objects and a certain container wall. It has been discovered that if only spatial relationships are available for design functions and if one wishes to package the objects tighfly to one end (or side) of the container, it is much better to maximize the sum of all separation distances from an opposite wall rather than minimize the sum of all separation distances with respect to the side desired. This is because during a minimization problem, the objective function is not as sensitive to changes in the separation distances (when objects rotate or translate) as it is when the objective function is maximized. The mass properties as design functions (discussed later) behave much better for this type of packaging.

Step 3: Optimal search for packaging problem.

PACKER is executed to search for the optimal placement of all objects in the container. This is where the problem is actually solved. The user reads in the problem file created by SETUP and geometry file created in NAVGRAPH. This step is basically the DESIGN module from the OPTDES package with the analysis function call being CALIPER. An optimization algorithm is chosen and the optimization software begins a search for the optimal design. The generalized reduced gradient has been found to be the best since it handles highly constrained problems with the best results. The algorithm runs a specified number of iterations or until an optimum is reached or assumed.

While in PACKER the user may also perform trial and error packaging with the SET UDV (set unscaled design variables) command. Here the rotations and translations may be modified for each object and PACKER will automatically inform the user if constraints were violated by the operation. Even if constraints are violated, the generalized reduced gradient algorithm can find feasible designs from a non-feasible starting position.

Step 4: Post processing.

At any iteration during the search for the optimal solution. the user may view the current state of the problem. The PRO POS command while in OPTDES puts the user into a NAVGRAPH menu where a DRAW command may be issued. All commands in the GLOBAL menu are available to the user (except those that modified the location and orientation of the objects). At any given iteration, the analysis variables are used to update the data base so that the correct view is drawn. When finished drawing, the user may exit or return to the OPTDES menu for further packaging work. If the user exits the NAVGRAPH menu, he can save the current design for a starting design of another optimization process.

Step 5: Further optimization.

The user can exit PACKER at any time and all design iterations are stored in a history file for later retrieval or continuation of an optimization process. Actually, the user has all options available that are detailed in the OPTDES users manual supplied with this report.

Even though the optimal packaging problem presented here in this section is useful, it is limited to packaging with the spatial relationships provided by the separation and/or interference distances from CALIPER. Many of the packaging problems would stop prematurely in local minima. This is mainly due to the multiple functions in the objective function. To remedy this, there needs to be analysis functions for optimal packaging that do not require multiply defined objective functions, such as, the mass properties as discussed in the next section.

PACKER with spatial relationships and mass properties.

In the previous section optimal packaging with just spatial relationships was discussed. This section discusses the optimal packaging software as enhanced to include the mass properties as possible design functions. There was no literature found in the engineering and computer graphics journals showing any related work that has been done in the area of optimally packaging objects into containers with mass properties as design functions. This section also discusses how to setup and run packaging problems with mass properties and compares this to packaging with just spatial relationships.

Much of the general discussion from the previous section applies here. The use of mass properties gives additional capabilities and options to the user for packaging objects into containers.

Geometric modeling convention

The method of representing 3-D arbitrarily shaped solid objects is the same as the geometry modeling convention which was used for CALIPER and PACKER, with spatial relationships as described in the previous two sections, with one additional step. Each object needs its mass properties calculated with MASSPROP in its initial design configuration.

Rotational degrees of freedom and the quaternion

The design variables remain the same, that is the translations and rotations (via the quaternion) for each object are defined as before. The discussion on rotations with quaternions is found in the previous section.

Mass properties for the packaging problem

The mass properties for each object and container in the initial design configuration are stored in a mass properties input file named massprop.MAS. The packaging software needs each object's $v, \rho, \bar{x}, \bar{y}, \bar{z}, I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}$, and I_{zy} (where ρ = density). The format of the massprop.MAS file is discussed in the users manual. Upon running PREPAC, the design's composite mass properties are calculated with the appropriate summation formula (as in equations 1 through 11 above except that the index i here represents the ith object rather than the ith element). The additional mass properties functions are then written to the OPTDES problem file for use in SETUP and PACKER.

As objects are translated and rotated during the optimal design search, the design functions change. The separation distances change as objects are moved (i.e. the design variables are changed) by OPTDES or the user. When an object is moved the design functions are recalculated. The separation distances are recalculated with another CALIPER call. The mass properties are updated by recalculation as follows.

The volume and mass remain constant because objects are not added and the container and objects do not change shape during an optimal packaging session.

The $\bar{x}, \bar{y}$ and $\bar{z}$ are recalculated with the following equations:

$$\bar{x} = \frac{\sum_i m_i x_i}{\sum_i m_i} \tag{23}$$

$$\bar{y} = \frac{\sum_i m_i y_i}{\sum_i m_i} \tag{24}$$

$$\bar{z} = \frac{\sum_i m_i z_i}{\sum_i m_i} \tag{25}$$

$$\tag{26}$$

Rotations do not affect $\bar{x}, \bar{y}$ and $\bar{z}$ because the quaternion rotates an object about a vector through the object's centroid.

The moments of inertia and products of inertia define a tensor (a 3X3 moment of inertia matrix). Therefore, the inertia terms are interdependent with respect to rotations. A rotation transformation matrix is needed to update each object's inertia tensor in the object's local coordinate system with respect to the global coordinate system. Each object's local coordinate system is initially parallel to the global coordinate system with its origin at the object's centroid. But as objects are rotated each object's inertia tensor with respect to the global coordinate system is changed. The rotations are in terms of a quaternion, therefore, we need a rotation transformation matrix from the quaternion design variables. The requisite transformation matrix **R** is given as:

$$\mathbf{R} = \begin{bmatrix} 2(e_0^2 + e_1{}^2) - 1 & 2(e_1 e_2 - e_0 e_3) & 2(e_1 e_3 + e_0 e_2) \\ 2(e_1 e_2 + e_0 e_3) & 2(e_0^2 + e_2{}^2) - 1 & 2(e_2 e_3 - e_0 e_1) \\ 2(e_1 e_3 - e_0 e_2) & 2(e_2 e_3 + e_0 e_1) & 2(e_0^2 + e_3{}^2) - 1 \end{bmatrix}$$

where

$$\begin{aligned} e_0 &= \cos\tfrac{\theta}{2} \\ e_1 &= n_1 \sin\tfrac{\theta}{2} \\ e_2 &= n_2 \sin\tfrac{\theta}{2} \\ e_3 &= n_3 \sin\tfrac{\theta}{2} \end{aligned}$$

and

$$e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1.$$

The moment of inertia matrix for each object in its new orientation is given with:

$$\mathbf{I} = \mathbf{R}\mathbf{I}'\mathbf{R}^{-1}.$$

The global inertia terms for the composite model are the sum of the individual transformed inertia terms via the parallel axis theorem.

The optimal packaging problem

The packaging problem is posed again as an optimization problem in the form:

Find:	rotations and translations for each object
Minimize or Maximize:	Some user defined objective function
Subject to:	Constraints that no object interfere with each other and that no object interferes with container walls. Also other optional constraints on mass properties.

The design variables in the packaging problem are still the three rotational and the three translational degrees of freedom per object. All design variables are applied to the object with respect to its original position. Objects are rotated via the quaternion about the quaternion's direction vector passing through the object's centroid.

The spatial constraint functions are still that the distance between objects and the distances between objects and container walls remain ≥ 0. Additionally, the mass properties allow more options to constrain the design to simulate real problems. For example, one may set equality constraints on the location of the center of gravity coordinates, or constrain the roll moment of inertia to be less than some value, etc..

The objective function now can be defined to minimize $\bar{y}$ to package objects to the bottom of a container. This has proven to be a much better definition since there is not the multiple function to one objective function mapping. Other possible objective functions are: 1) minimize or maximize $\bar{x}$ or $\bar{z}$ or some combination, 2) minimize roll, pitch or yaw moments of inertia (this tends to package the object toward the centroid), 3) minimize the products of inertia (this tends to place the objects in a fashion that approaches symmetry) and 4) some combination of the above with care to apply proper coefficients in the multiple objective function mapping for proper scaling in the optimization.

The following steps are taken to solve a packaging problem with mass properties.

Step 1: Create and initialize a starting design.

The user creates the initial design in NAVGRAPH by modeling the objects and the container, and then by translating and rotating the objects in NAVGRAPH into their intended initial position.

Step 2: Calculate mass properties.

Each object needs its mass properties supplied. This can be done with several methods. One way is to mesh each object with solid finite elements and run MASSPROP for each object. Another way is to use existing data if measured experimentally, or previously with MASSPROP. Each object's mass properties need to be input into the file massprop.MAS. The massprop.MAS file is read by PREPAC and PACKER at a user prompt.

Once the model has been created, it needs to be initialized by running PREPAC. PREPAC calculates all separation distances between all objects and between all objects and container walls, and then initializes the values of total volume, total mass, composite $\bar{x}$, $\bar{y}$, $\bar{z}$, and the moments of inertia about the local coordinate system defined by a parallel axis system with its origin at ($\bar{x}$, $\bar{y}$, $\bar{z}$). PREPAC then writes an OPTDES problem file containing all initial analysis variables and analysis functions for input into SETUP and eventually PACKER.

Step 3: The Setup.

To setup the packaging problem, one executes the SETUP module of OPTDES that reads in the problem file from PREPAC then defines design variables, constraint functions, and the objective functions for the desired optimization problem.

The design variables are defined as a subset of the analysis variables. If design variables are not defined, the object to which the analysis variables belong will not move in the direction of the design variables omitted. The user can use this to his advantage in problem setup to keep objects from moving with respect to certain degrees of freedom (e.g. let objects translate but not rotate, etc.). Upper and lower bounds are placed on the variables. These bounds are important as they greatly effect the scaling of the problem. By varying the bounds on the rotations and translations it is possible to favor one over the other. This becomes important in highly constrained problems where there are tight fits between objects and the container wall.

Again, the separation distances should be constrained to be greater than some very small positive value to save interference distance calculations when the solution is near the optimal design. If packaging material is needed around the objects, the constraint limits should reflect the necessary thickness of the material.

The objective function is chosen here from the design functions defined by the mass properties. It has been discovered that if one wishes to package the objects as tightly to one end (or side) of the container, it is much better to maximize or minimize the coordinates of the global composite centroid ($\bar{x}, \bar{y}, \bar{z}$, or some combination) instead of the sum of all separation distances from a container wall. This is because the optimization function is at most a mapping of only several mass properties functions rather than a multitude of separation distances. An objective function defined by minimization or maximization of $\bar{x}$, for example, is much more sensitive to translations and rotations of objects than any objective function defined with spatial relationships alone.

Step 4: Optimal search for packaging problem.

PACKER is executed to search for the optimal placement of all objects in the container. This is where the problem is actually solved. The user reads in the problem file created by SETUP, the geometry file created in NAVGRAPH, and the massprop.MAS file. This step is basically the DESIGN module from the OPTDES package with the analysis function module being CALIPER. An optimization algorithm is chosen and the optimization software begins a search for the optimal design. The generalized reduced gradient has been found to be the best since it handles highly constrained problems with the best results. The algorithm runs a specified number of iterations or until an optimum is reached or assumed.

While in PACKER the user may also perform trial an error packaging with the SET UDV (set unscaled design variables) command. Here the rotations and translations may be modified for each object and PACKER will automatically inform the user if constraints were violated by the operation. The DIS DV and DIS DF command can be used to view the values of the design variables and design functions at any iteration. Even if constraints are violated, the generalized reduced gradient algorithm can find feasible designs from a non-feasible starting position.

Step 5: Post processing.

Again, at any iteration during the search for the optimal solution, the user may view the current state of the problem. The PRO POS command while in OPTDES puts the user into a NAVGRAPH menu where a DRAW command may be issued. All commands in the GLOBAL menu are present to the user (except those that modify the location and orientation of the objects). At any given iteration, the analysis variables are used to update the data base so that the correct view is drawn. When finished drawing, the user may exit or return to the OPTDES menu for further packaging work.

Step 6: Further optimization.

The user can exit PACKER at any time and all design iterations are stored in a history file for later retrieval or continuation of an optimization process. Actually, the user has all options available that are detailed in the OPTDES users manual supplied with this report.

The mass properties give much improvement over just spatial relationships in the packaging process. They help the search routines search for the optimal solution faster by supplying a function that does not need multiple mapping in the objective function. It also gives the user much more latitude in objective function definition.

Examples of packaging problems solved with PACKER

Several examples are presented here showing optimal packaging problems that were solved with PACKER.

The first example, shown in Figure 7, has a container with a flat bottom and straight sides. Three cylinders are placed in a feasible starting design where there are no interferences initially. The middle cylinder is oriented to a position not parallel with the other cylinders. The objective function was defined to minimize $\bar{y}$. The design variables were so that each cylinder could rotate and translate freely in all directions. The spatial constraints were defined so that there would be at least .01 inches between all cylinders and between all cylinders and container walls. The GRG algorithm was employed, and after just seven iterations, the optimal positions were found (see Figure 7).

A second example is seen in Figure 8. This shows a hemi-spherical shell as a container wall. Several objects, namely two cylinders, two pyramids and one long box, are placed in positions defining an initial feasible design. Each object is free to rotate and translate in all directions. The objective function was to minimize $\bar{y}$. The optimal solution was reached after 16 GRG iterations with no user intervention.

The above problems would have been solved in less iterations if better starting designs were supplied, however, the robustness of PACKER is apparent when optimal solutions are found regardless of the starting design.

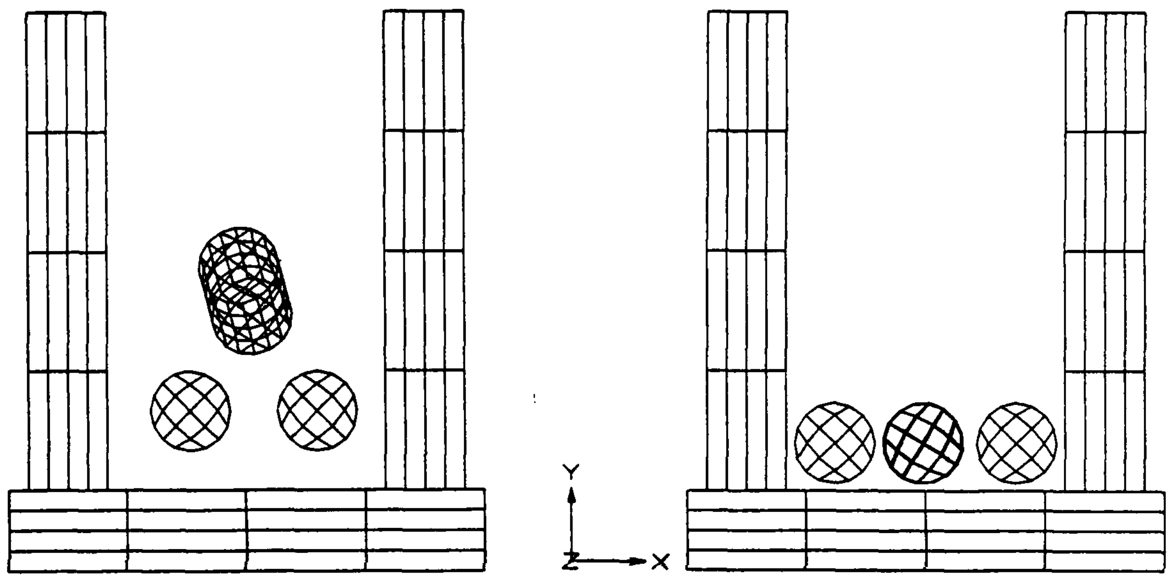


Figure 7: An example of PACKER packaging three cylinders into a container. The objective was to minimize $\bar{y}$.

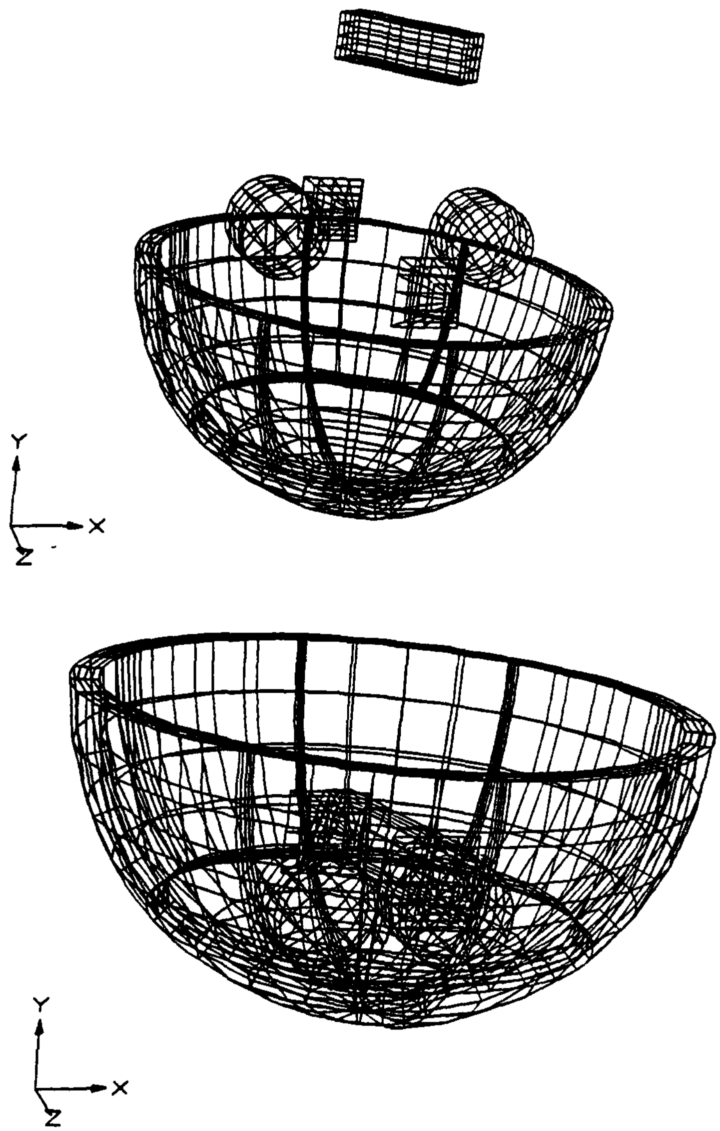


Figure 8: An example of PACKER packaging several objects into a hemi-spherical container. The objective was to minimize $\bar{y}$.

Capabilities added to increase speed, robustness and flexibility.

There were several tasks (in addition to those discussed in the previous sections) that were completed to increase the robustness, flexibility and efficiency of the overall CALIPER and PACKER software. A discussion of the additional tasks is given here.

Explicit gradients.

The algorithms in OPTDES require derivatives of analysis functions with respect to analysis variables. Without user supplied derivative software explicitly, OPTDES calculates derivatives numerically using the finite forward difference approach:

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(\Delta x_j + x_j) - f_i(x_j)}{\Delta x_j}$$

where x_j is the jth design variable and f_i is the ith design function. With this method, each design variable is perturbed in turn by a small amount in order to evaluate

$$f_i(\Delta x_j + x_j).$$

This requires n analysis calls to evaluate one gradient call at the beginning of each design iteration where n is equal to the number of design variables in the problem.

In the optimal packaging problem, the finite forward difference would require an analysis call for every rotation and translation design variable for every object in the model to get just one gradient call. It was readily seen that if we had explicit gradient formulae we could save an enormous amount of computations (analysis calls). Recall that each analysis call involves computing all distances between all objects and between all objects and all container walls (a cpu intensive call) along with the mass properties of the composite model. Explicit gradients of the analysis functions involved in the optimal packaging problem were derived. The resulting algorithms were programmed into the ANAGRA subroutine in OPTDES for PACKER. The following is a discussion of these formulae. The calculation begins by getting the separation distances and interference distances between all objects and between all objects and container walls (just one analysis call) and stores the end points of each distance calculated. Object one is defined as the object with the design variable being considered in the gradient calculation (i.e. the object that would be perturbed if forward difference was applied) and object two is defined as the object with which the separation distance is calculated.

Separation/Interference distances with respect to translation.

A unit vector $\mathbf{v}$ is defined by the direction of the design variable begin considered (i.e. for x translation $\mathbf{v} = (1, 0, 0)$). The end points of each distance defines a vector $\mathbf{s}$. The direction of $\mathbf{s}$ points away from object one and toward object two. The value of the gradient for separation distance between object one and object two with respect to translation is found by

$$\mathbf{v} \cdot \mathbf{s}.$$

Separation/Interference distances with respect to rotation.

A vector $\mathbf{r}$ is defined as the vector pointing from the centroid of object one to the separation distance end point on the surface of object one. The vector $\mathbf{v}$ is defined by the direction of the design variable considered (i.e. for rotation about the x axis $\mathbf{v} = (1, 0, 0)$). Using $\mathbf{r}$, $\mathbf{v}$ and $\mathbf{s}$, the gradient for separation with respect to rotation is found by

$$(\mathbf{v} \times \mathbf{r}) \cdot \mathbf{s}.$$

Center of gravity coordinates with respect to translation.

The gradients of the three coordinates of the center of gravity with respect to translations are calculated by differentiating the summation formula for each c.g. coordinate separately.

For the gradient of $\bar{x}$ with respect to an x translation of an object, we start with

$$\bar{x} = \frac{\sum_i m_i \bar{x}_i}{\sum_i m_i}$$

expanded it becomes

$$\bar{x} = \frac{1}{m_{tot}}(m_1\bar{x}_1 + m_2\bar{x}_2 + \cdots + m_k\bar{x}_k + \cdots + m_i\bar{x}_i).$$

The partial derivative with respect to x_k (where k represents the the design variable being considered) we get

$$\frac{\partial \bar{x}}{\partial x_k} = \frac{m_k}{m_{tot}}.$$

The partial derivative of $\bar{x}$ with respect to translation in the y or z directions is zero. This means that translation in y or z has no effect on the gradient of $\bar{x}$.

Similar expressions are obtained for $\bar{y}$ and $\bar{z}$.

Center of gravity coordinates with respect to rotation.

The gradients of the three coordinates of the center of gravity with respect to any rotation is zero in all cases since we rotate about the centroid of each object. The formulae for $\bar{x}, \bar{y}$ and $\bar{z}$ do not depend on the rotational degrees of freedom.

Moments of Inertia with respect to translation.

The gradients of the moments of inertia with respect to translation are calculated by differentiating the summation formula for each moment of inertia component separately with respect to the design variable being considered.

For the gradient of I_{xx} with respect to a y translation, we start with

$$I_{xx} = \sum_i m_i(\bar{y}_i^2 + \bar{z}_i^2)$$

expanded it becomes

$$I_{xx} = m_1(\bar{y}_1^2 + \bar{z}_1^2) + \cdots + m_k(\bar{y}_k^2 + \bar{z}_k^2) + \cdots + m_i(\bar{y}_i^2 + \bar{z}_i^2).$$

The partial derivative with respect to y_k (where k represents the design variable being considered) we get

$$\frac{\partial I_{xx}}{\partial y_k} = 2m_k\bar{y}_k.$$

With respect to z_k we get

$$\frac{\partial I_{xx}}{\partial z_k} = 2m_k\bar{z}_k.$$

With respect to x_k we get

$$\frac{\partial I_{xx}}{\partial z_k} = 0.$$

Similar expressions are obtained for I_{yy}, I_{zz}, I_{xy}, I_{xz} and I_{yz}.

Moments of Inertia with respect to rotation.

Obtaining the explicit gradients of the moments of inertia with respect to rotation (where the rotations are in terms of the quaternion) is very complex and requires many computations. It was decided to use the finite forward difference method for this case with a $\Delta x, \Delta y$ or $\Delta z = .01$ perturbation. The only term in the summation formulae for I that changed, with respect to a rotation design variable, is the term involving the design variable being considered. Only the term involving the design variable is modified according to the quaternion transformation matrix associated with the perturbation. The gradient is then found by the usual formula:

$$\frac{\partial I_{xx}}{\partial y_k} = \frac{I_{xx}(\Delta y_k + y_k) - I_{xx}(y_k)}{\Delta y_k}.$$

Similar equations for the other terms can be derived. This formula turned out to be a rather elegant algorithm as programmed in the ANAGRA subroutine and has shown to work well.

NAVGRAPH group commands.

Several capabilities were necessarily added to NAVGRAPH to facilitate the operations of CALIPER and PACKER. Most of the added capabilities were in the GLOBAL

GROUP menu to allow the user the freedom to manipulate the solid objects into the desired models for physical fit studies and optimal packaging problems. The users manual shows the syntax of each command. These commands are discussed here.

GLOBAL GROUP TRANSLATE allows the user to translate an object to a new position or replicate an object with the translation. The user is queried on group names and the x, y, z translation.

GLOBAL GROUP ROTATE allows the user to rotate an object to a new position or replicate an object with the rotation. The user is queried on group names and the x, y, z rotation.

GLOBAL GROUP SCALE allows the user to scale an object to a new size or to replicate an object with the scale values given. The user is queried on group names and x, y, z scale values.

GLOBAL GROUP QROTATE allows the user to rotate an object to a new position or replicate an object with the quaternion representation of rotation. The user is queried on group names, vector of rotation, and magnitude of rotation about the vector.

GLOBAL GROUP SNAP allows the user to translate an object to a new position with a snap-to command. The user is queried for two group names and two points (one for each group). The second group is snapped to the first group so that the two given points occupy the same point.

The GLOBAL GROUP CALIPER command calculates the separation and interference distances between all objects or optionally between just two objects. The group names are printed out along with the separation and/or interference distances.

Once the separation and interferences have been calculated the GLOBAL GROUP HIGHLIGHT will display all interferences as vectors with flashing lines. The color of the flashing lines is supplied by the user.

GLOBAL GROUP DISTANCE allows the user to get the distance between any two points on any two objects. The user is queried on group names and point numbers.

The ability to use the local hardware options on the Tektronix 4107 (and 4109) has been made available with the GLOBAL DISPLAY SEGMENT command. This command toggles on and off for segments to be captured during a NAVGRAPH display. Once the segment has been captured the local 2-D zoom and pan may be used on the Tektronix 4107 (and 4109).

The ability to use the local hardware options on the Tektronix 4129 has been made available with the GLOBAL DISPLAY HARDWARE command. This command toggles on and off for segments to be captured during a NAVGRAPH display. Once the segment has been captured the local 3-D zoom, pan, rotate, etc. may be used on the Tektronix 4129.

The capability to use Tektronix 4957-4958 digitizers was added to the GLOBAL DIGITIZER menu. This allows easy point input and other entity picking and deleting. The digitizer hook-up and initialization needs the following Tektronix SET UP mode commands made to insure proper function of the digitizer board (the 4958).

PASSIGN PX:4958
PBAUD PX:9600
PBITS PX:1 7
PPARITY PX:ODD
PFLAG PX:NONE
PEOF PX:NONE

NAVGRAPH ployfil text font.

The text font available in NAVGRAPH was simple line text only, and the user had no control over location and orientation. NAVGRAPH was enhanced to include a polyfil font. The user now has control over size, slant, location, orientation and color of the text in NAVGRAPH. This new font is much more readable and allows the user the freedom to label the various parts of a model as needed.

Methodologies to create AEROHEAT shapes with NAVGRAPH.

The NUMERICAL FLOW FIELD PROGRAM FOR AERODYNAMIC HEATING ANALYSIS (AEROHEAT) as described in AFFDL-TR-85-3054 has the ability to solve aero heating predictions to several exact geometry aero-shapes as well as surfaces generated from curve fitting routines. It was discovered that the geometric input data, required by the AERO-HEATING program, was so different from the geometric data produced by NAVGRAPH, that it made the development of a translator between NAVGRAPH and AEROHEAT highly impractical. However, most of the exact geometry shapes defined in the AEROHEAT users manual (namely, the tangent ogive, ogive cylinder, cone with nose cap, cylinder with nose cap, and hyperboloid) were modeled in NAVGRAPH. Figure 9 shows several of these as modeled in NAVGRAPH. The methodologies used to create these shapes are documented in the tutorial in Part 2.

Fast surface meshing capability.

A command to mesh many surfaces in one command was included in the MESH CREATE ELEMENT menu. This allows the user to easily create a finite element mesh on all surfaces in the data base for use in visualization with solid smooth renderings.

MOVIE.BYU software supplied.

The MOVIE.BYU software is supplied with a Tektronix 4129 driver that was modified to load the video lookup table with a 256 color ramp in one color. This gives the user the ability to display a very smooth rendering of solid polygonal models that look realistic. The basic color may be changed without a need to redraw. The user is also prompted for background and text colors that may be different than the ramp color. MOVIE.BYU offers transparencies, multiple light sources, animation and

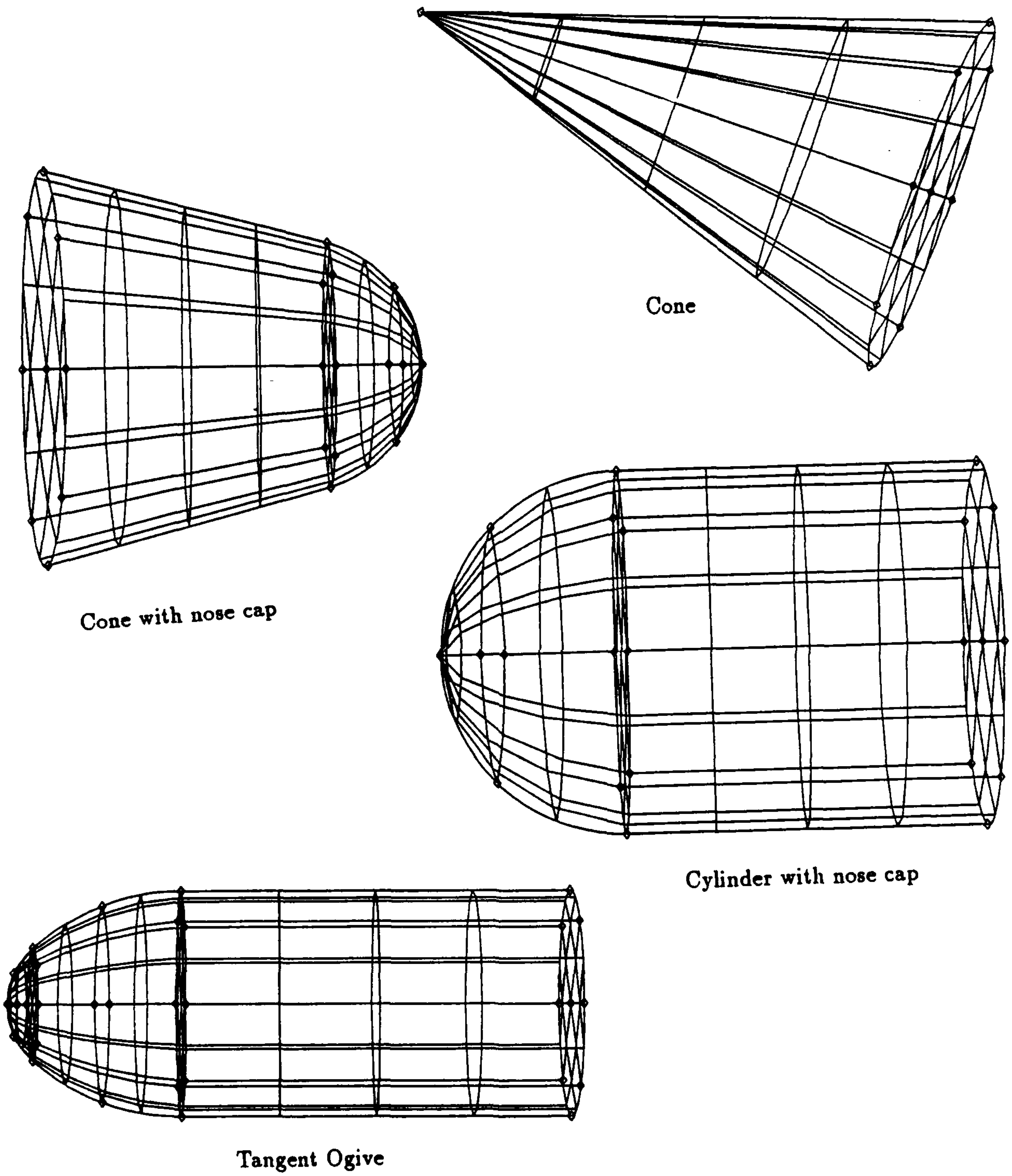


Figure 9: NAVGRAPH models of several Aeroheat shapes.

other capabilities not yet available in NAVGRAPH. However, NAVGRAPH can produce smooth renderings of finite element models.

GRG algorithm modifications.

The generalized reduced gradient (GRG) algorithm used in CALIPER was modified to be more robust and efficient for the specific problem of solving separation and interference calculations. The modifications are given here.

The resetting of the Hessian matrix after n (n being the number of design variables) iterations was removed. This change allows the GRG algorithm to save function and constraint information to help search for the solution near the optimum. This change helped to improve the robustness for problems with very shallow and narrow objective functions.

The value of EPSOPT was changed from 0.001 to 0.0001. This allows for an acceptable convergence in some separation calculations that previously were not detecting interferences properly. This increases computation time slightly but increases robustness a great deal.

The parabolic fit routine was modified to find the minimum point in a separation calculation when the minimum point is between the first and second point on the parabola.

A last ditch mode was incorporated for the interference calculation to see if the equality constraints are violated and sets a flag if they are. This enables GRG to enter a Newton-Raphson method to satisfy the constraints and start the optimization process with a valid starting design. This eliminates a few function calls each time an interference iteration is started at an infeasible design point.

Solid groupings into objects.

As discussed throughout this report, a very important and beneficial capability was added to NAVGRAPH to allow the user to group a number of solids together to model complex, convex and non-convex objects. This is simply the boolean add function as found in CSG.

Packaging by layers.

The capability to package by layers was studied during this effort. There was no method implemented in software to automatically package by layer, but a straight forward heuristic approach is offered here.

Step one: Model the desired container wall with solids and group them into one object.

Step two: Model the desired objects by creating the necessary solids. Group the solids into the desired objects. Limit the number of objects to a reasonable number (e.g. the number that would fill the first layer).

Step three: Initialize the design with PREPAC and setup the desired optimal packaging problem with SETUP such that the objects are packaged toward the desired container wall.

Step four: Run PACKER and optimally package the objects. This can be done as outlined in the PACKER section of this report. Use the GRG algorithm or manually set the unscaled design variables such that the objects are packaged as tightly as possible toward the container wall.

Step five: Once the objects are packaged tightly, exit PACKER with a NAVGRAPH exit via the PRO POS display option. This will save the current design as a NAVGRAPH database with updated object positions.

Step six: Execute NAVGRAPH with the updated data base file from step five. Ungroup all of the solids in the objects (leaving the container alone) with the GLOBAL GROUP DELETE command. Then group all solids (previously defined in the original objects) into one large group. This group may then be replicated with the GLOBAL GROUP TRANSLATE command with the appropriate translation. The appropriate translation would come from inspection or by using the CALIPER command to detect the interference distance between the first new object and the second. Repeat this step until the container is full or until the desired number of objects are packaged.

Other heuristic methods and trouble shooting.

During the development and testing of CALIPER, PACKER and the data base creation, there were some activities found in each area that worked better than others. A list of heuristical 'rules of thumb' and trouble shooting guides were complied and included in the users manual/tutorial.

Translator Software Development

In order for the software package to be most useful, it must be able to generate various data base files which can be read by other existing programs. This capability is called forward translation and allows a user to transfer finite element and/or geometry information to other programs for further development and/or analysis. Similarly, the software package must be able to read information generated by other programs. This capability is called reverse translation and allows a user to read data that was created, altered, or analyzed by another program.

Several forward and reverse translators were developed under this work. A user interface was written to facilitate the execution of these translators in a fashion that resembles the NAVGRAPH user interface. All translators that were developed are described here.

TEKNICAD To NAVGRAPH Translator.

A reverse translator was written to format TEKNICAD data files into a NAVGRAPH data base file. TEKNICAD is a 2-D drafting program marketed by Tektronix. The TEKNICAD to NAVGRAPH translator converts all points, straight lines (those with one line segment per line) and circular arcs into like NAVGRAPH geometry entities. The user then may use the points, lines and arcs in a NAVGRAPH session for further model development.

Translators Between NAVGRAPH and VUMADM.

The Missile Aerodynamic Design Method (MADM) system calculates aerodynamic characteristics of complex configurations at supersonic and hypersonic speeds. This is the next generation supersonic/hypersonic aerodynamic analysis code with major upgrades to SHABP (MARK IV). The pre- and post-processor software (VUMADM) can prepare geometry data for use in analysis as well as display analysis results.

The VUMADM forward translator developed here writes geometry and finite element information generated in NAVGRAPH (including nodes, elements, points, parametric cubic lines and circular arcs) to a VUMADM format file. This file is read by the VUMADM module and can then be displayed, modified further, or otherwise prepared for analysis by the MADM program.

The VUMADM reverse translator reads a VUMADM format file containing geometry and finite element information and writes it to a NAVGRAPH data base file. The user is prompted for the VUMADM file name and the program takes a few seconds to several minutes to translate the file depending on the model size. A translated VUMADM file, containing the points and lines defining the F-15 nose and canopy, is shown in Figure 10. The finite element mesh, also translated is shown. A diagnostic file containing information about the entities translated, entities unsupported and any errors that occurred is also written and is found on disk with a .DIA extension to the

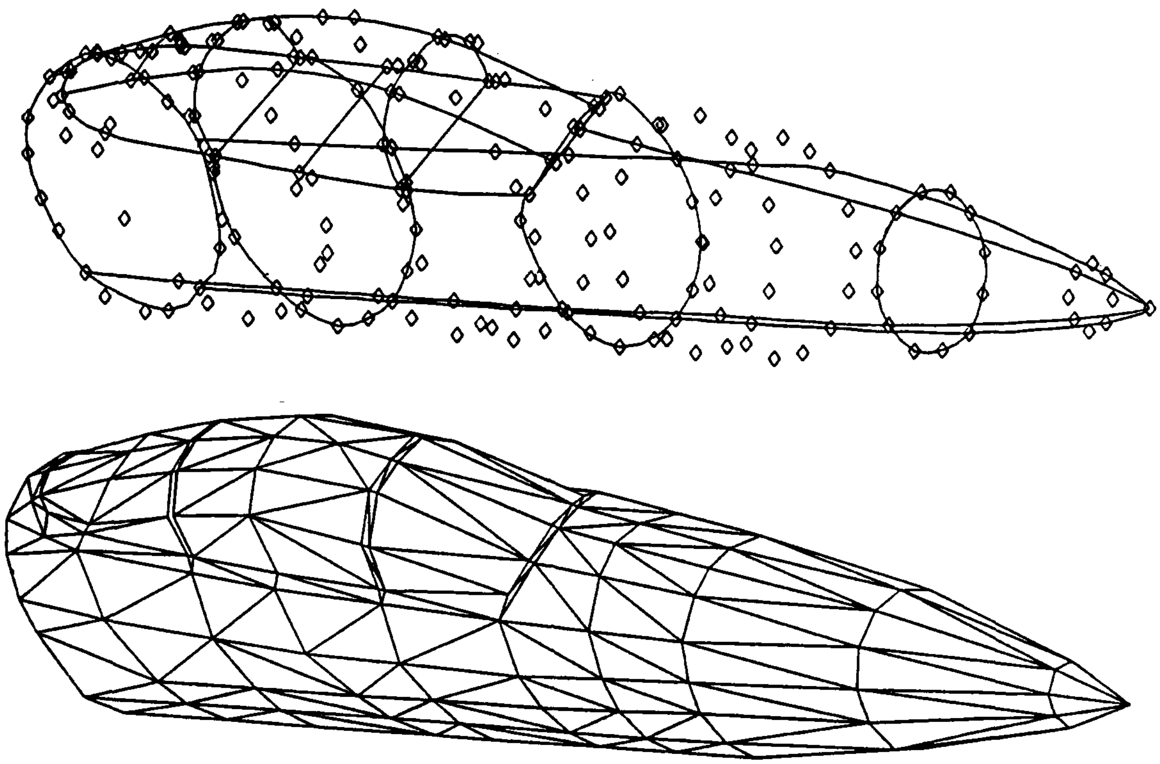


Figure 10: A VUMADM model of the F-15 nose geometric and finite element models read into the NAVGRAPH data base.

file name input above.

IGES Translators Between NAVGRAPH and ANVIL-5000.

ANVIL is a program that can develop numerical controlled milling machine instructions from geometric surface definitions. ANVIL can read IGES (Initial Graphics Exchange Specifications) neutral files containing geometric points, lines, splines and circular arcs developed by other CAD/CAM systems. Because ANVIL (like other CAD/CAM programs) uses the IGES neutral file for data translation, having an IGES translator in NAVGRAPH is important. Both forward and reverse IGES translators were developed.

The forward translator developed here writes geometric information generated in NAVGRAPH such as points, lines, splines and circular arcs to IGES Version 3.0 format. This translator, made up of several subroutines, represents a major modification to the previous IGES translator that was available in NAVGRAPH. This translator

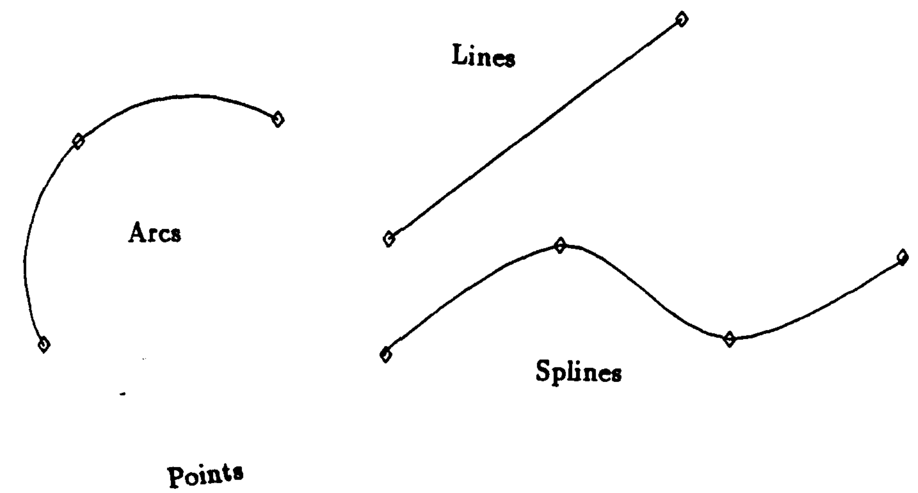


Figure 11: NAVGRAPH model via the IGES translator.

requests the name of the IGES file to be output and the units of the model (i.e. inches, meters, etc.). A list of current global section parameters then appears on the screen for acceptance by default or for selective editing. The global section of an IGES file contains information describing the preprocessor and information needed by the postprocessor to handle the file. Upon acceptance of the global section parameters, the IGES file is written and the program returns to the forward translation menu. If an error occurs in opening the IGES file, an error message is given and returns to the forward translation menu.

Note that the IGES parametric spline entity (entity 112) definition written by this translator is slightly modified in order that the program ANVIL-5000 can interpret it correctly. The definition, however, is sufficiently general and can be read by other programs with IGES translators.

The IGES reverse translator developed reads geometric information (points, lines, splines and circular arcs) from an IGES Version 3.0 file to a NAVGRAPH data base file. The user inputs the name of the IGES file to be read when prompted. The time required to translate the file is dependent on the system used and the size of the model. Large models can take several minutes to translate. Figure 11 is a NAVGRAPH plot of several points, lines, parametric cubic lines, and arcs translated from an IGES file.

The program keeps track of the type and number of entities translated, entities unsupported in NAVGRAPH and any errors that occurred in translation. This information is written to a diagnostic file with the extension .IGR on the input file name. Examination of the diagnostic file can aid in reproducing lost entities or checking for errors.

Note that points associated with straight lines are written twice to the NAVGRAPH data base since the IGES definition of a line is given by its endpoints. The endpoints of spline segments and points that make up arc segments are only written once.

Once a NAVGRAPH file has been translated to an ANVIL file, one can use the data defining the lines to create surfaces and instruct ANVIL to create the numerical machining instructions to manufacture the desired aero-shape.

Translator for AFATL/ASE

The AFATL/ASE format is the required format for input files to such codes as radar signature prediction programs used by the U.S. Air Force. The forward translator developed writes the finite element information generated in NAVGRAPH (nodal coordinates and element connectivity) to the AFATL/ASE format. The resulting file contains 1) target components (logical group of triangular facets or elements), 2) nodal point coordinates, and 3) element connectivity (or link-lists). The subroutine named NAV2ASE writes two output files. One file is an unformatted (or binary) file containing the above mentioned information. The name of this file is entered by the user when prompted for by the program. The second file is a formatted (ASCII) file which contains a list of component numbers and the beginning and ending element numbers associated with each component. The name of this file is the same as the entered file name except with a .TRG extension which identifies it as a target component list file. If an error is encountered in opening either file, an error message is sent to the screen and the program returns to the forward translation menu.

Note that for every skip in element numbers found in the NAVGRAPH data base a new component is formed and written to the AFATL/ASE format file. Only linear triangular surface elements are accepted by AFATL/ASE.

The AFATL/ASE format reverse translator reads an AFATL/ASE format file (containing nodal and triangular element information) into storage arrays from which component information is selectively extracted and rewritten to a NAVGRAPH data base file. Component information may be seen in the default filename.TRG file which is associated with the AFATL/ASE format data base file. The program requests the name of the file to be read and then prompts the user for the beginning and ending component numbers to be translated. A message is sent to the screen indicating that a particular component is being translated. As with the forward translator appropriate error messages are sent to the screen if errors are found in opening files. Since a single node can be referenced several times for a group of adjacent elements, a simple technique was devised to write each node only once. Each time a node is placed in the NAVGRAPH data base its number is placed in an array. Before the next node is written, it is checked against existing nodes in the array. If present, the node is not written to the data base, otherwise it is. This saves considerable time because the data base routines sort the existing data each time they are called

It is important to note that the coordinates defining nodes are also written as points to the NAVGRAPH data base. This allows a user to access these points (which are geometric entities) in NAVGRAPH for the creation of surface and solid entities. The surfaces can then be remeshed with different elements or the solid meshed with solid hexahedrons elements. The surfaces, solids and new meshes are then available for a wide range of analysis and/or further development.

The U.S. Air Force typically has models of aircraft and missiles in the AFATL/ASE data base. These models can now be translated to NAVGRAPH using this reverse translator. The user may define geometry entities in NAVGRAPH. Once the geometry entities are defined in NAVGRAPH, they may be used in compatibility studies and/or optimal packaging problems. This is the method used in generating the F-15 model in the next section. Note that this does require user interaction, and may require much time and expertise in using the points supplied by the AFATL/ASE data base to define the necessary lines, surfaces and solid entities. Several 'tricks-of-the-trade' and other guidelines found to work using this procedure are given in the tutorial.

Data Base Generation with NAVGRAPH.

A data base consisting of an F-15 aircraft with equipment, several weapons and submunitions was generated with NAVGRAPH for test cases in 1) optimal packaging of submunitions in missiles, 2) studying the physical fit compatibility of stores on aircraft, 3) finite element model generation and 4) data base format translation.

Aircraft Data Base Generation.

After some discussion of which aircraft to model, and how to model it, it was jointly decided by Eglin AFB and APTEK that the preferred method would be to use an F-15 model with data reformatted from the AFATL/ASE data base. This was accomplished using the ASE to NAVGRAPH translator and reading in all nodes and elements from the F-15 ASE data file into NAVGRAPH. Each node was also defined as a point for use in defining lines, surface and solids in NAVGRAPH.

It was necessary to fit the points to parametrically defined lines and surfaces by hand with NAVGRAPH line and surface commands. Solids were then defined with lines and surfaces where appropriate.

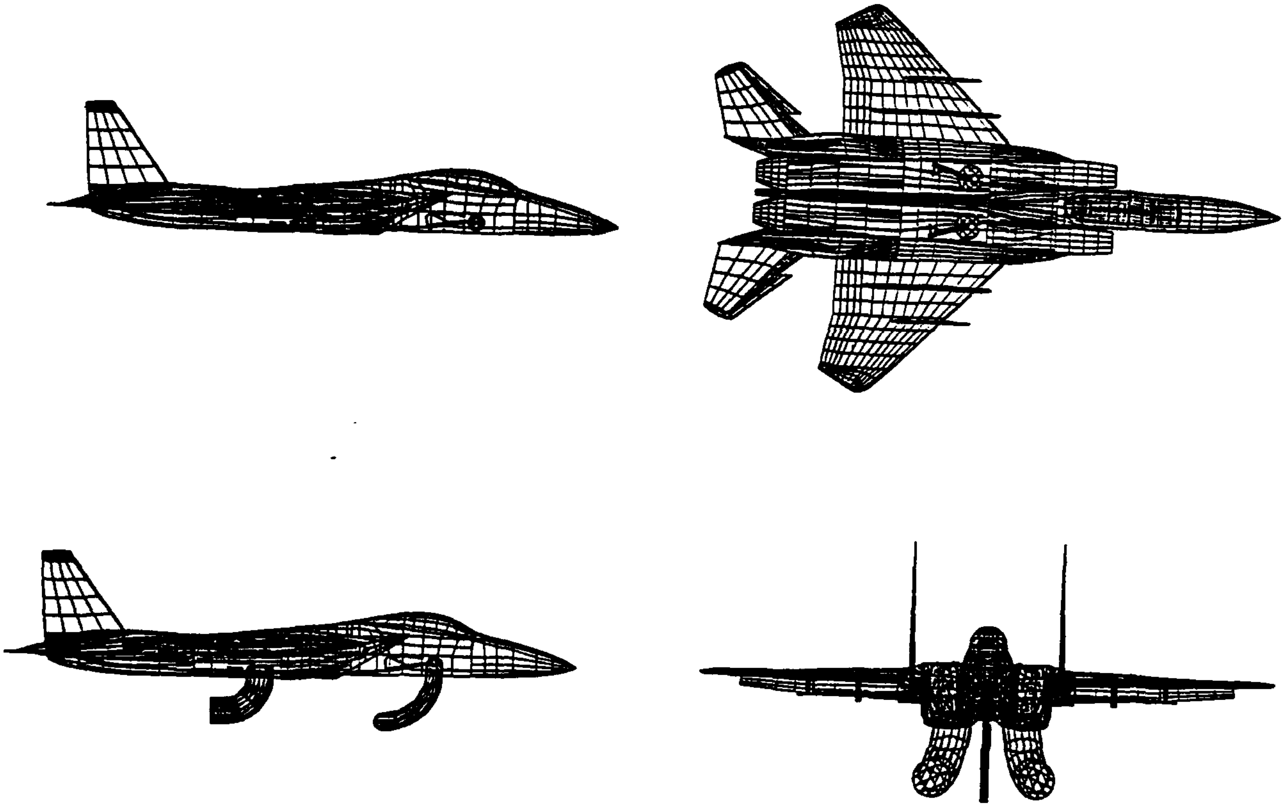


Figure 12: F-15 'clean' and 'swept' configurations.

NAVGRAPH with CALIPER can now be used, with the F-15 model and weapons, to perform physical fit studies of loadouts.

Two models of the F-15 were created, one with its equipment and gear in a 'clean' nominal flight configuration and another with its equipment and gear in a 'swept' configuration (see Figure 12). The 'swept' configuration has the flaps and landing gear modeled with NAVGRAPH by sweeping the flap and landing gear solids through the deployed articulation.

Because the F-15 aircraft was modeled, the following equipment was also modeled.

- MAU-12 Bomb Rack
- LAU-117 Rail Launcher
- 610-gal. Fuel Tank

Figure 13 displays several of these models.

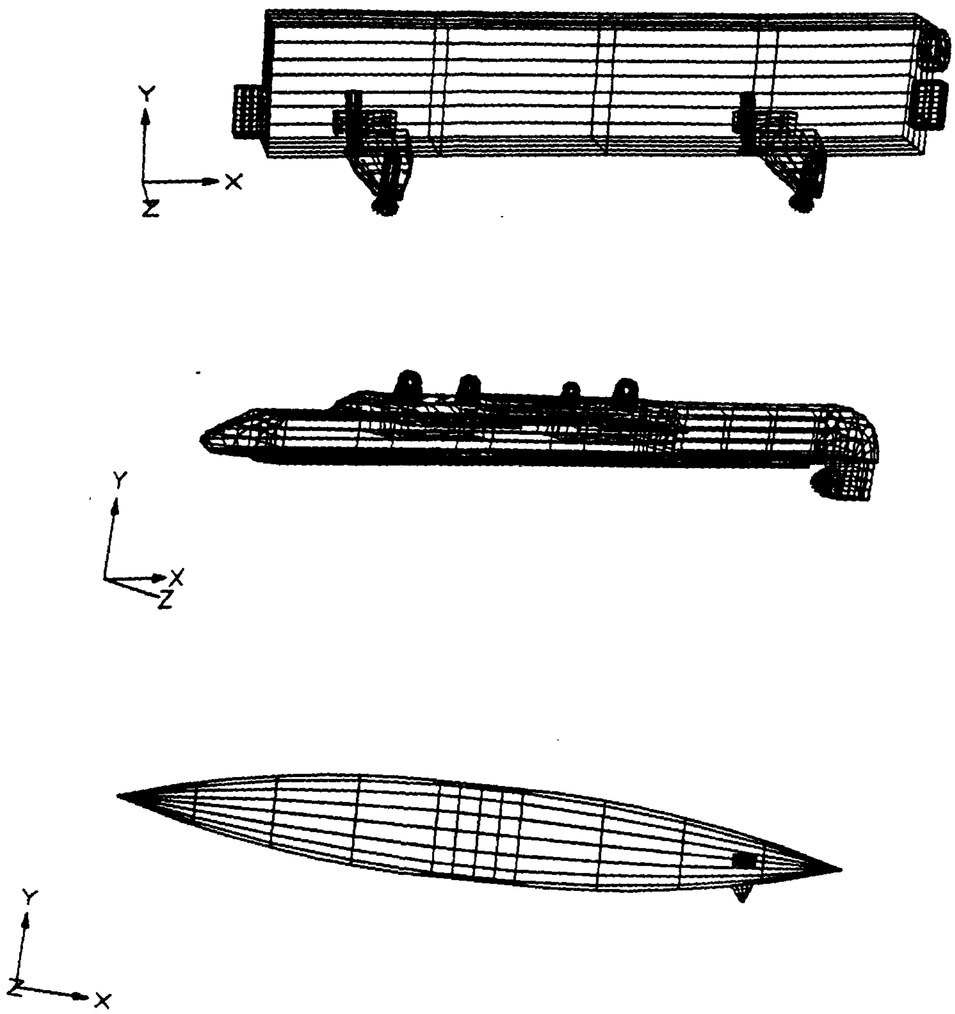


Figure 13: NAVGRAPH models of the MAU-12 rack, LAU-117 rail launcher and 610-gal fuel tank for the F-15 aircraft.

Weapon Data Base Generation.

NAVGRAPH was used to generate a geometric model of the external geometry of each of the following weapons. This data base was created in sufficient detail to determine store-to-store or store-to-aircraft separation or interference.

- MK-82 1380901
- MK-84 ASIM
- BLU-109 8463195
- MAVERICK AGM65 ASIM
- SUU-64/65 777108
- Sparrow ASIM
- AIM-9 ASIM

Figure 14 shows the geometric models of several of the above weapons.

The SUU-64/65 *submunition dispenser* was modeled with its internal geometry also, in sufficient detail for optimal packaging of submunitions. Models containing fins both deployed and stowed were created.

NAVGRAPH was used to generate two models of the external geometry of each of the following weapons. One file contains the weapon with its wings extended, and the other file contains the weapon with its wings retracted. This data base was also created with sufficient detail to determine the store to store or store to aircraft separation and/or interference. Since the following weapons are modular in nature, their modularity was exploited for economy.

- GBU-10C/B
- GBU-10G/H/J/
- GBU-12B/B
- GBU-22
- GBU-24
- GBU-24A/B

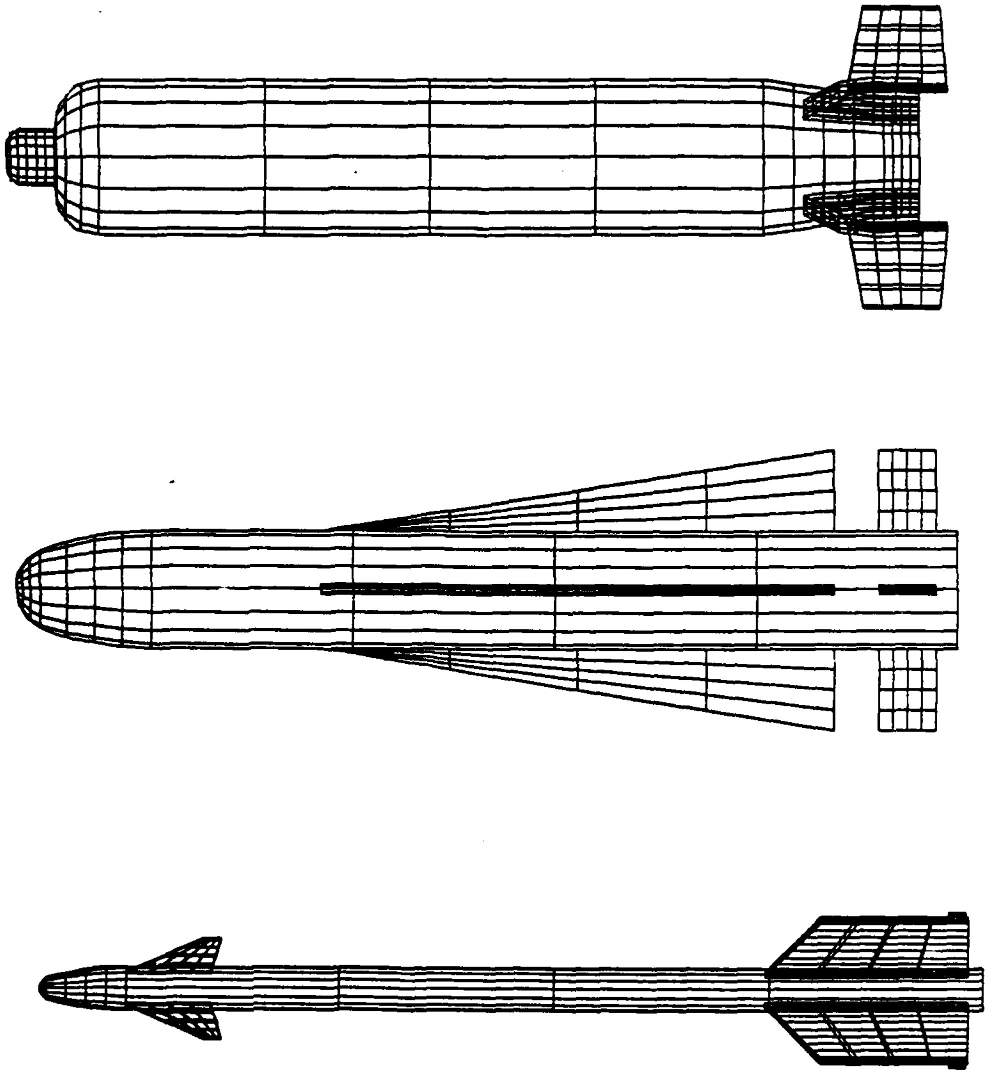


Figure 14: NAVGRAPH models of the SUU-64/65, MAVERICK and AIM-9 missile.

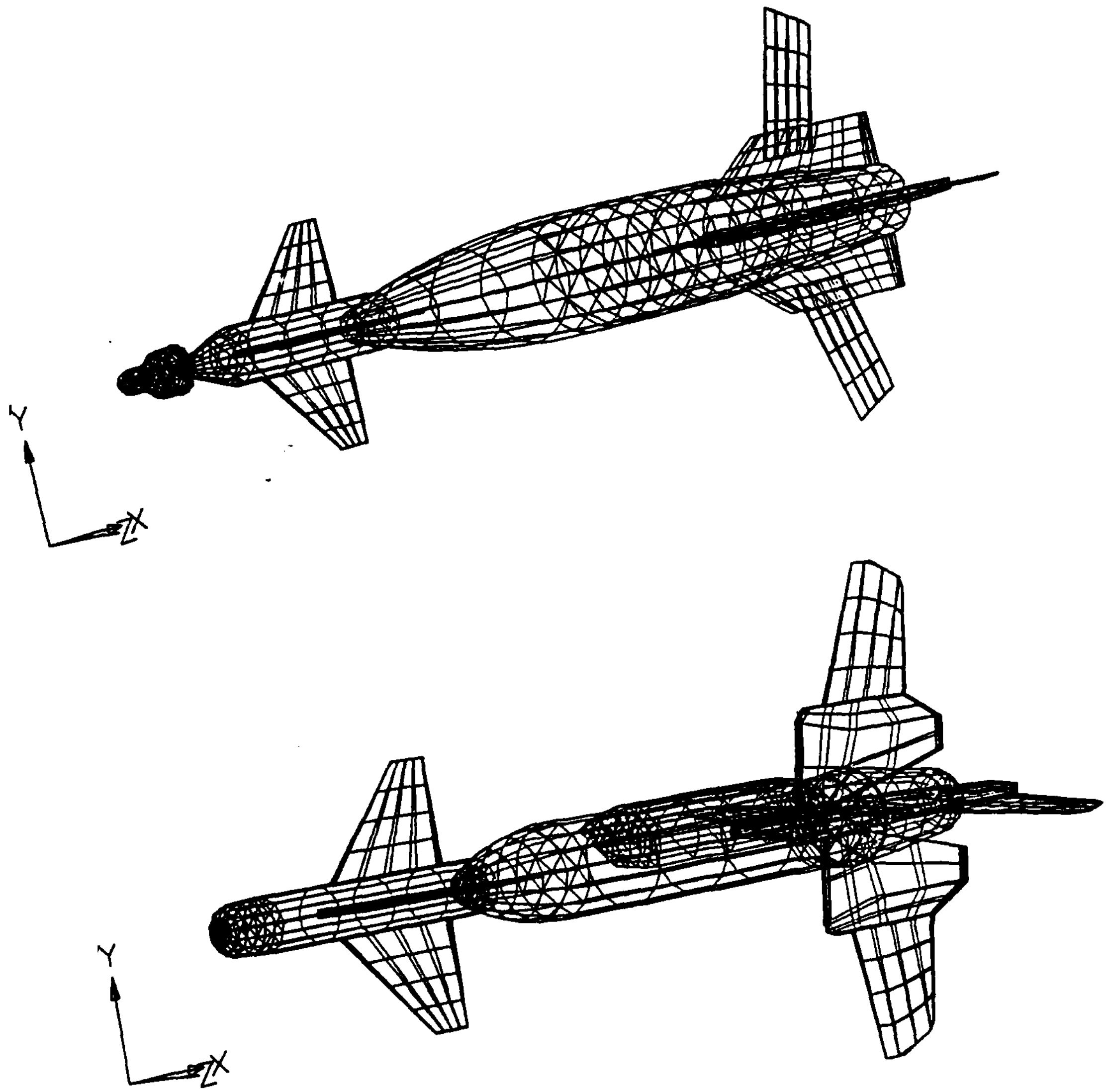


Figure 15: NAVGRAPH models of the GBU-10C/B and the GBU-24-A/B.

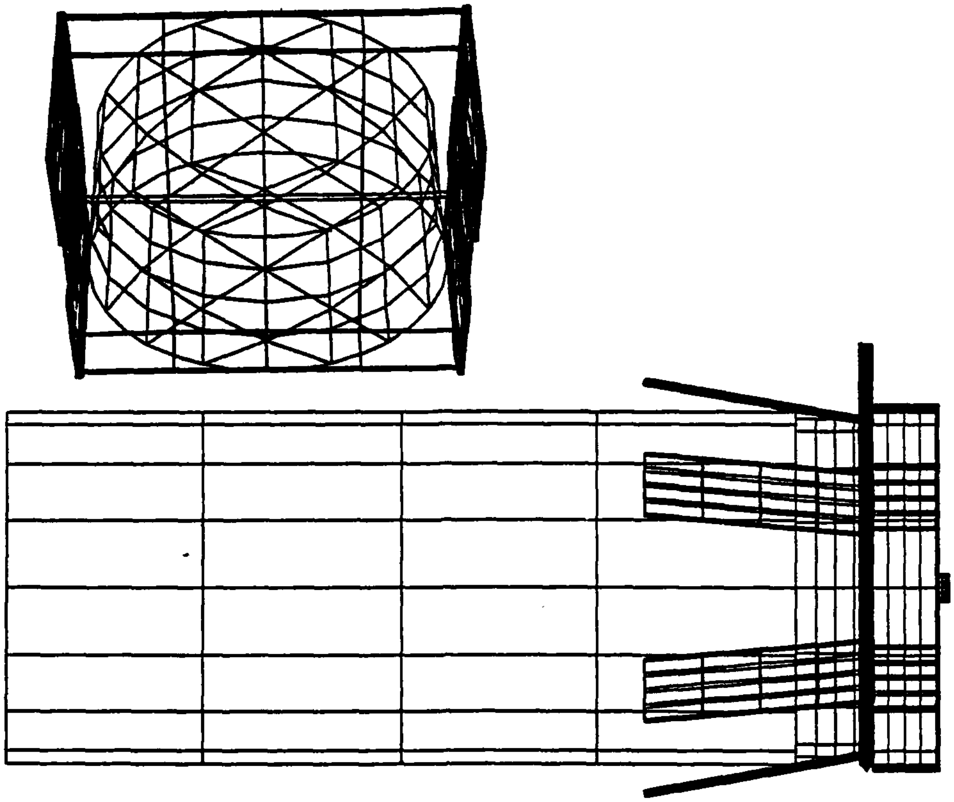


Figure 16: NAVGRAPH models of the GATOR and CEB submunitions.

Figure 15 shows the geometric models of several of the weapons mentioned above.

Submunition Data Base Generation.

NAVGRAPH was also used to generate a data base of the following submunitions. The data base was created to provide sufficient detail to determine the submunition to submunition and submunition to dispenser separation and interference distances for optimal packaging. Two models, one detailed and one simple, of the GATOR, the CEB and the BLU 63 are provided. The detailed models contain all of the curved, complex geometry (see Figure 16). The simple models contain a minimum of detail needed for the optimal packaging problem.

- BLU-92 GATOR 9299710
- BLU-97 CEB X809420
- BLU-63 68C6001
- BLU-106 BKEP 162-72168
- HTW PENETRATOR SK5433186

Summary.

The work completed and documented here has described the software tools that were developed and delivered to help the Air Force weapon designer take a weapon from concept to test specimen quickly and accurately. The capabilities include:

1. Creation of geometric models of aircraft, weapons, submunitions, pylons, racks, etc. to the necessary detail.
2. Recall geometric models from the an existing data base to define complex, convex and non-convex shapes for a desired model.
3. Study the physical fit compatibility of stores on aircraft.
4. Calculate the mass properties (including surface area, volume, mass, center of gravity and moments of inertia) of the desired model.
5. Optimally package submunitions and/or internal components into a dispenser missile cargo bay with constraints on mass properties.
6. Generate finite element models of geometric models for use in prediction software codes.
7. Format geometric data for use by numerically controlled milling machines to machine test specimens.
8. Accept geometric and finite element data from other Air Force data base formats.
9. And generate computer color graphic displays (line drawings or smooth rendering) of the models including interference highlighting.

This section has discussed the technical aspects of the work completed under the contract. The next section contains the users manual and tutorial for all software modules developed under the contract.

Part 2

The Users Manual and Tutorial

The Users Manual

This section contains the specific command syntax for all new capabilities added to NAVGRAPH and the Eglin data base translators. These commands supplement the users manuals for both NAVGRAPH and OPTDES (which accompany this users manual).

The file named MSTAR.COM (a VMS DCL command file) is the user friendly menu selector for the many executable modules of the software package. Note that MSTAR stand for MOVIESTAR which is another name (the commercial name) for NAVGRAPH. Upon issuing the @MSTAR.COM command, the following will appear on the screen:

```
Select a NAVGRAPH option:

     1. NAV09          Run NAVGRAPH TEK4109 (4208)
     2. NAV29          Run NAVGRAPH TEK4129
     3. FOR(MAT)       Run NAVGRAPH Eglin Data Formatter
     4. PRE(PAC)       Run PREPAC (CONVEN)
     5. SETUP          Run SETUP
     6. PAC09          Run PACKER TEK4109(DESIGN)
     7. PAC29          Run PACKER TEK4129(DESIGN)
     8. CAL(IPER)      Run CALIPER
     9. MASS(PROP)     Run MASS PROPERTIES
    10. DBM(ERGE)      Run DATABASE MERGE
    11. DIR(ECTORY)    Local Directory Listing
    12. DCL            Enter DCL Command
    13. Q(UIT)         Quit NAVGRAPH

Option
```

One simply enters the desired option and the chosen module is executed. Every module is available to execute at this time. You can run DCL command from inside MSTAR.COM as well as look at the directory.

Upon exiting MSTAR.COM, the following banner is displayed. Care was taken programming the numerous algorithms that make up this software package. Also, there were many test cases run to verify correct execution of all programs. However, there is no assurance ' ıat there are no bugs, but we feel that they are minor. If there are any bugs found, please report them.

```
****************************************
*                                      *
*  Report bugs to:                     *
*    Mark Landon                       *
*    APTEK Inc.                        *
*    1257 Lake Plaza Dr.               *
*    Colorado Springs, CO 80906-3578   *
*    (719) 576-8100                    *
*                                      *
****************************************
```

The new NAVGRAPH commands are given in the next 21 pages. They are followed by the Eglin data base translator commands.

MENU:
GLOB
1. CLEAR
2. COLOR =>
3. DEFAULTS =>

4. DIGITIZER =>

5. DISPLAY-OPTIONS =>
6. DRAW
7. ERASE =>
8. EXIT
9. GROUP =>
10. HARDCOPY =>
11. HELP
12. LABEL =>
13. LIST =>
14. MENU
15. PLOT =>
16. RECALL
17. RESET
18. TIME
19. VIEW-OPTIONS =>

ENTER: -> 4 <CR> or DIG <CR>

KEYBOARD: GLOB DIGI

DESCRIPTION:

The DIGITIZER command selects the DIGITIZER menu.

MENU: GLOB.DIGI

ENTER: -> 1 <CR> or DET <CR>

KEYBOARD: GLOB DIGI DETE

DESCRIPTION:

The DETECT command toggles ON and OFF to enable or disable graphic input from a device such as a digitizing tablet, mouse, or thumbwheels. The default value is OFF.

PROMPTS:

Enter on or off (default = off)>

EXAMPLES:

Enter on or off (default = off)> ON

Enables graphic input.

Enter on or off (default = off)> OFF

Disables graphic input.

MENU: GLOB.DIGI

ENTER: -> 2 <CR> or OR <CR>

KEYBOARD: GLOB DIGI ORIE

DESCRIPTION:

The ORIENTATION command creates a transformation between screen coordinates and world coordinates. This is needed to properly orient a drawing on a digitizing tablet or some other graphic input device. The user will be prompted to detect any three points using the graphic input device and to enter the corresponding world coordinates (x and y) for each point.

PROMPTS:

Detect point1 on digitizer
Enter point1 (x,y)>

Detect point2 on digitizer
Enter point2 (x,y)>

Detect point3 on digitizer
Enter point3 (x,y)>

EXAMPLES:

Detect point1 on digitizer
Enter point1 (x,y)> 0.0 0.0

Detect point2 on digitizer
Enter point2 (x,y)> 10.0 0.0

Detect point3 on digitizer
Enter point3 (x,y)> 5.0 20.0

Creates 2-d transformation between screen and drawing using the three point 0.0,0.0 10.0,0.0 5.0,20.0.

MENU: GLOB.DIGI

ENTER: -> 3 <CR> *or* Z- <CR>

KEYBOARD: GLOB DIGI Z-VA

DESCRIPTION:

The Z-VALUE command is used to enter a constant z-value for points enter with a graphic input device.

PROMPTS:

Enter constant z value
(default = 0.00000E+00)>

EXAMPLES:

Enter constant z value
(default = 0.00000E+00)> 5.5

A z coordinate of 5.5 will be given to every point entered with a graphic input device until a new z value is given.

MENU:
GLOB.DIGI
1. DETECT
2. ORIENTATION
3. Z-VALUE

4. DEVICE =>

ENTER: -> 4 <CR> or DEV <CR>

KEYBOARD: GLOB DIGI DEVI

DESCRIPTION:

The DEVICE command selects the DEVICE menu.

MENU: GLOB.DIGI.DEVI

ENTER: -> 1 <CR> or MO <CR>

KEYBOARD: GLOB DIGI DEVI MOUS

DESCRIPTION:

The MOUSE command identifies a mouse as the graphic input device.

MENU: GLOB.DIGI.DEVI

ENTER: -> 2 <CR> or TA <CR>

KEYBOARD: GLOB DIGI DEVI TABL

DESCRIPTION:

The TABLET command identifies a digitizer tablet as the graphic input device.

MENU: GLOB.DIGI.DEVI

ENTER: -> 3 <CR> or TH <CR>

KEYBOARD: GLOB DIGI DEVI THUM

DESCRIPTION:

The THUMBWHEELS command identifies the thumbwheels or joydisk as the graphic input device.

MENU: GLOB.DISP

ENTER: -> 26 <CR> or TEX <CR>

KEYBOARD: GLOB DISP TEXT

DESCRIPTION:

The TEXT command prompts the user for text information. Text can be used for comments dimensions, etc.. There are two text fonts, polygonal and line. The text is two dimensional but can be manipulated in three dimensions. When maniplating text in 2-d, screen coordinates (0 to 1) are used for positioning. When manipulating text in 3-d, world coordinates are used. Only one type of text (polygonal or line) can be displayed.

PROMPTS:

```
    Polygonal or Line Text? [ P or (L) ]
    >
    Text Message of Text String X
    ( ):
    >
    Width (1.0) and Height (1.0) :>
    2-D or 3-D Text Position (2):>
if 2-D
    Position X ( 0.00), Y ( 0.00), Angle ( 0.00) :>
if 3-D
    Position Origin X ( 0.00), Y ( 0.00), Z ( 0.00) :>
    Position Vector X ( 0.00), Y ( 0.00), Z ( 0.00) :>
    Angle to Slant Text:
    (default = 0.00000E+00)>
Additional Prompts:
    Color Red (1.0), Blue (1.0), Green (1.0) :>
    Modify Text String X ? (default = no)>
    Add Another Text String ? (default = no)>
    Delete Existing Text Strings ? (default = no)>
    Modify Existing Text Strings ? (default = no)>
    Show Summary of Existing Text Strings ? (default = yes)>
    Text String Number to be Modified :>
```

MENU: GLOB.DISP.ENTI

ENTER: -> 17 <CR> or TEX <CR>

KEYBOARD: GLOB DISP ENTI TEXT

DESCRIPTION:

The TEXT command toggles ON and OFF and turns the display of text on or off.

PROMPTS:

Enter on or off (default = off)>

EXAMPLES:

Enter on or off (default = off)> ON

Turns display of text on.
Enter on or off (default = off)> OFF

Turns display of text off.

MENU: GLOB.LIST

ENTER: -> 16 <CR> or TEX <CR>

KEYBOARD: GLOB LIST TEXT

DESCRIPTION:

The TEXT lists all existing text strings on the screen.

MENU: GLOB.DISP

ENTER: -> 25 <CR> or HA <CR>

KEYBOARD: GLOB DISP HARD

DESCRIPTION:

The HARDWARE-GRAPHICS turns the local hardware graphics capabilities on and off.

PROMPTS:

Enter on or off (default = off)>

EXAMPLES:

Enter on or off (default = off)> ON

Turns hardware graphics on.

Enter on or off (default = off)> OFF

Turns hardware graphics off.

MENU: GLOB.DISP

ENTER: -> 27 <CR> or SE <CR>

KEYBOARD: GLOB DISP SEGM

DESCRIPTION:

The SEGMENT turns the local segment graphics capabilities on and off.

PROMPTS:

Enter on or off (default = off)>

EXAMPLES:

Enter on or off (default = off)> ON

Turns segments on.

Enter on or off (default = off)> OFF

Turns segments off.

MENU: GLOB.GROU

ENTER: -> 9 <CR> or CA <CR>

KEYBOARD: GLOB GROU CALI

DESCRIPTION:

The CALIPER command executes the CALIPER distance calculator.

PROMPTS:

Calculate all interferences ? (default = no)>
Input the first group name (<CR> to quit):>
Input the second group name:>
Do you want to specify interference direction ? (default = no)>
Input x,y,z components of interference direction vector:>

EXAMPLES:

Calculate all interferences ? (default = no)>
AVAILABLE GROUP NAMES ARE:
G1
G2
Input the first group name (<CR> to quit):> G1
Input the second group name:> G2
Do you want to specify interference direction ? (default = no)> Y
Input x,y,z components of interference direction vector:> 1 0 0
Distance information will be listed here.

MENU: GLOB.GROU

ENTER: -> 10 <CR> or DIS <CR>

KEYBOARD: GLOB GROU DIST

DESCRIPTION:

The DISTANCE command calculates the distance between two points.

PROMPTS:

Enter the first point >
Enter the second point >

EXAMPLES:

Enter the first point > 5
Enter the second point > 10

Distance between two points is : 20.0

MENU: GLOB.GROU

ENTER: -> 11 <CR> or HI <CR>

KEYBOARD: GLOB GROU HIGH

DESCRIPTION:

The HIGHLIGHT-INT command turns the interference highlights on and off. This feature is availible on Tektronix machines. A highlight consists of a blinking line along the vector of interference. The CALIPER command must be executed to calculate interferences whenever changes are made to existing groups.

PROMPTS:

Highlight line color components (red, green, blue) ? >
Do you want to list interferences? (default = no)>

EXAMPLES:

Highlight line color components (red, green, blue) ? > 1 1 0
Interferences will be drawn on the screen.
Do you want to list interferences? (default = no)> y
Interference information will be listed.

MENU: GLOB.GROU

ENTER: -> 12 <CR> or RO <CR>

KEYBOARD: GLOB GROU ROTA

DESCRIPTION:

The ROTATE command rotates a group using x, y, and z rotations. A group can be rotated with or without creating a new group.

PROMPTS:

Enter new group name
(default =)>
Enter enter group to rotate
(default =)>
Enter rotation angle (x,y,z) >

EXAMPLES:

To create a new group:

Enter new group name
(default = group1)> group2
Enter enter group to rotate
(default = group2)> group1
Enter rotation angle (x,y,z) > 10 15 20

To rotate a group:

Enter new group name
(default = group1)> group1
Enter enter group to rotate >
(default = group1)> group1
Enter rotation angle (x,y,z) > 20 5 20

MENU: GLOB.GROU

ENTER: -> 13 <CR> or QR <CR>

KEYBOARD: GLOB GROU QROT

DESCRIPTION:

The QROTATE command rotates a group using a quaternion. A quaternion consists of a vector and a rotation angle about that vector. A group can be rotated with or without creating a new group.

PROMPTS:

```
Enter new group name
(default = )>
Enter enter group to rotate
(default = )>
Enter the center point of the group (x,y,z) >
Enter the rotation vector (x,y,z) >
Enter rotation angle (alpha) >
```

EXAMPLES:

To create a new group:

```
Enter new group name
(default = group1 )> group2
Enter enter group to rotate
(default = group2 )> group1
Enter the center point of the group (x,y,z) > 5,10,0
Enter the rotation vector (x,y,z) > 1,0,0
Enter rotation angle (alpha) > 20
```

To rotate a group:

```
Enter new group name
(default = group1 )> group1
Enter enter group to rotate
(default = group1 )> group1
Enter the center point of the group (x,y,z) > 3,0,1
Enter the rotation vector (x,y,z) > 1,1,1
Enter rotation angle (alpha) > 30
```

MENU: GLOB.GROU

ENTER: -> 14 <CR> or SN <CR>

KEYBOARD: GLOB GROU SNAP

DESCRIPTION:

The SNAP command is another method for translating a group. A point is selected in each of two groups. The second group is then translated so that the world coordinates of the point in the second group are the same as the point in the second first.

PROMPTS:

Enter first group name >
Enter a point on first group >
Enter second group name >
Enter a point on second group >

EXAMPLES:

Enter first group name > group1
Enter a point on first group > 10
Enter second group name > group2
Enter a point on second group > 50

MENU: GLOB.GROU

ENTER: -> 15 <CR> or TR <CR>

KEYBOARD: GLOB GROU TRANS

DESCRIPTION:

The TRANSLATE command translates a group using x, y, and z translations. A group can be translated with or without creating a new group.

PROMPTS:

Enter new group name
(default =)>
Enter enter group to translate
(default =)>
Enter translation (x,y,z) >

EXAMPLES:
To create a new group:
Enter new group name
(default =)> group2
Enter enter group to translate
(default =)> group1
Enter translation (x,y,z) > 5,0,10.5
To translate a group:
Enter new group name
(default =)> group1
Enter enter group to translate
(default =)> group1
Enter translation (x,y,z) > 3.0,0,0

MENU: GLOB.GROU

ENTER: -> 16 <CR> or SC <CR>

KEYBOARD: GLOB GROU SCAL

DESCRIPTION:

The SCALE command scales a group using x, y, and z scale factors. A group can be scaled with or without creating a new group.

PROMPTS:

Enter new group name
(default =)>
Enter enter group to scale
(default =)>
Enter scale factors (x,y,z) >

EXAMPLES:

To create a new group:

Enter new group name
(default =)> group2
Enter enter group to scale
(default =)> group1
Enter scale factors (x,y,z) > .9,1,1

To scale a group:

Enter new group name
(default =)> group1
Enter enter group to scale
(default =)> group1
Enter scale factors (x,y,z) > .75,.5,.3

Eglin AFB Translators

The Eglin AFB tranlators module of NAVGRAPH translates data base files to and from AFATL/ASE, IGES, TEKNICAD, and VUMADM formats. (NOTE: The AFATL/ASE format is used for radar signature prediction; the VUMADM format is required for use in the Missile Aerodynamic Design Method (formerly S/HABP) system.)

The VUMADM translator provides for both geometry and finite element entity translation, the AFATL/ASE translator for finite element entity translation, and the IGES and TEKNICAD translators for geometry entity translation only. The IGES forward translator features selective editing of the global section parameters the control file processing.

EGLIN AFB TRANSLATORS.FORWARD

MENU:

EGLIN AFB TRANSLATORS
  1. **FORWARD** **=>**
  2. REVERSE =>

ENTER: -> 1 <CR or FO <CR>

KEYBOARD: FORW

DESCRIPTION:

The FORWARD command selects the FORWARD translation menu.

MENU:

FORWARD
1. **AFATL/ASE**
2. IGES
3. TEKNICAD
4. VUMADM

ENTER: -> 1 <CR> or AF <CR>

KEYBOARD: FORW AFAT

DESCRIPTION:

The AFATL/ASE command creates an AFATL/ASE format file from the finite element information (nodal coordinates and element connectivity) of the NAVGRAPH database. Target components, node coordinates, and element link-lists are written to an AFATL/ASE compatible file.

PROMPTS:

Enter output file name
(default = problem name)>

EXAMPLES:

Enter output file name
(default = problem name)> PYLON.ASE

Creates the AFATL/ASE format file PYLON.ASE from the finite element information in the NAVGRAPH database.

MENU:

FORWARD
1. AFATL/ASE
2. **IGES**
3. TEKNICAD
4. VUMADM

ENTER: -> 2 <CR> or IG <CR>

KEYBOARD: FORW IGES

DESCRIPTION:

The IGES command creates an IGES file from the geometry of the NAVGRAPH database. Points, lines, splines, and circular arcs are written to an IGES compatible file.

PROMPTS:

Enter units
(default = INCH)>

Enter output file name
(default = problem name)>

EXAMPLES:

Enter units
(default = INCH)> MM

Enter output file name
(default = problem name)> NEWFILE.IGE

Creates the IGES file NEWFILE.IGE from the geometric information in the NAVGRAPH database and gives it units of millimeters.

MENU:

FORWARD
1. AFATL/ASE
2. IGES
3. **TEKNICAD**
4. VUMADM

ENTER: -> 3 <CR> or TE <CR>

KEYBOARD: FORW TEKN

DESCRIPTION:

The TEKNICAD command creates a TEKNICAD file from the geometry of the NAVGRAPH database. 2-D points, lines, splines, and circular arcs are written to a TEKNICAD compatible file.

PROMPTS:

Enter output file name
(default = problem name)>

EXAMPLES:

Enter output file name
(default = problem name)> PLATE.TEK

Creates the TEKNICAD file PLATE.TEK from the geometric information (points, lines, splines and circular arcs) in the NAVGRAPH database.

MENU:

FORWARD
1. AFATL/ASE
2. IGES
3. TEKNICAD
4. **VUMADM**

ENTER: -> 4 <CR> or VU <CR>

KEYBOARD: FORW VUMA

DESCRIPTION:

The VUMADM command creates a VUMADM format file from the geometry and finite element information of the NAVGRAPH database. Points, lines, splines, circular arcs, nodes, and elements are written to a VUMADM compatible file.

PROMPTS:

Enter output file name
(default = problem name)>

EXAMPLES:

Enter output file name
(default = problem name)> MISSLE.MAD

Creates the VUMADM file MISSLE.MAD from the geometric and finite element information in the NAVGRAPH database.

MENU:

```
  EGLIN AFB TRANSLATORS
     1.  FORWARD          =>
     2.  REVERSE          =>
```

ENTER: -> 2 <CR or RE <CR>

KEYBOARD: REVE

DESCRIPTION:

The REVERSE command selects the REVERSE translation menu.

MENU:

REVERSE
1. **AFATL/ASE**
2. IGES
3. TEKNICAD
4. VUMADM

ENTER: -> 1 <CR> or AF <CR>

KEYBOARD: REVE AFAT

DESCRIPTION:

The AFATL/ASE command translates an AFATL/ASE format file containing finite element information to a NAVGRAPH database. Nodal coordinates and element link-lists are translated from AFATL/ASE format to NAVGRAPH format.

PROMPTS:

Enter input file name
(default = problem name)>

EXAMPLES:

Enter input file name
(default = problem name)> PYLON.ASE

Copies the finite element and nodal coordinate information from the AFATL/ASE file, PYLON.ASE, to the current NAVGRAPH database.

MENU:

REVERSE
1. AFATL/ASE
2. **IGES**
3. TEKNICAD
4. VUMADM

ENTER: -> 2 <CR> or IG <CR>

KEYBOARD: REVE IGES

DESCRIPTION:

The IGES command translates an IGES file to a NAVGRAPH database. Points, lines, splines, and circular arcs are translated from IGES to NAVGRAPH format.

PROMPTS:

Enter input file name
(default = problem name)>

EXAMPLES:

Enter input file name
(default = problem name)> NEWFILE.IGE

Copies the geometric information (points, lines, splines and circular arcs) from the IGES file, NEWFILE.IGE, to the current NAVGRAPH database.

MENU:

REVERSE
1. AFATL/ASE
2. IGES
3. **TEKNICAD**
4. VUMADM

ENTER: -> 3 <CR> or TE <CR>

KEYBOARD: REVE TEKN

DESCRIPTION:

The TEKNICAD command translates a TEKNICAD file to a NAVGRAPH database. 2-D Points, lines, splines, and circular arcs are translated from TEKNICAD format to NAVGRAPH format.

PROMPTS:

Enter input file name
(default = problem name)>

EXAMPLES:

Enter input file name
(default = problem name)> PLATE.TEK

Copies the geometric information (2-D points, lines, splines and circular arcs) from the TEKNICAD file, PLATE.TEK, to the current NAVGRAPH datbase.

MENU:

REVERSE
1. AFATL/ASE
2. IGES
3. TEKNICAD
4. **VUMADM**

ENTER: -> 4 <CR> or VU <CR>

KEYBOARD: REVE VUMA

DESCRIPTION:

The VUMADM command translates a VUMADM format file to a NAVGRAPH database. Points, circular arcs, parametric cubic lines, nodes, and elements are translated from VUMADM format to NAVGRAPH format.

PROMPTS:

```
Enter input file name
(default = problem name)>
```

EXAMPLES:

```
Enter input file name
(default = problem name)>  MISSLE.MAD
```

Copies the geometric (points, circular arcs, and parametric cubic lines), nodal coordinate, and finite element information from the VUMADM file, MISSLE.MAD, to the current NAVGRAPH database.

The Tutorial

The tutorial presented here is a collection of actual runs with the various modules of the software package.

First, we give several NAVGRAPH modeling examples. Because every module needs a geometric model to work with, it is important that the user becomes very good at using NAVGRAPH. The more the user learns about the commands and capabilities of NAVGRAPH, the better and faster he will be able to solve the problem at hand. Among the examples given are the basic exact geometry shapes as found in the AEROHEAT users manual. Also, the steps to create a MAVERICK missile are provided. These examples are in the navgraph.RUN format. They can be typed line-by-line interactively. They can also be run automatically with the RECALL command in NAVGRAPH.

There are other modeling examples throughout that the user may glean some pointers on how to model better and faster.

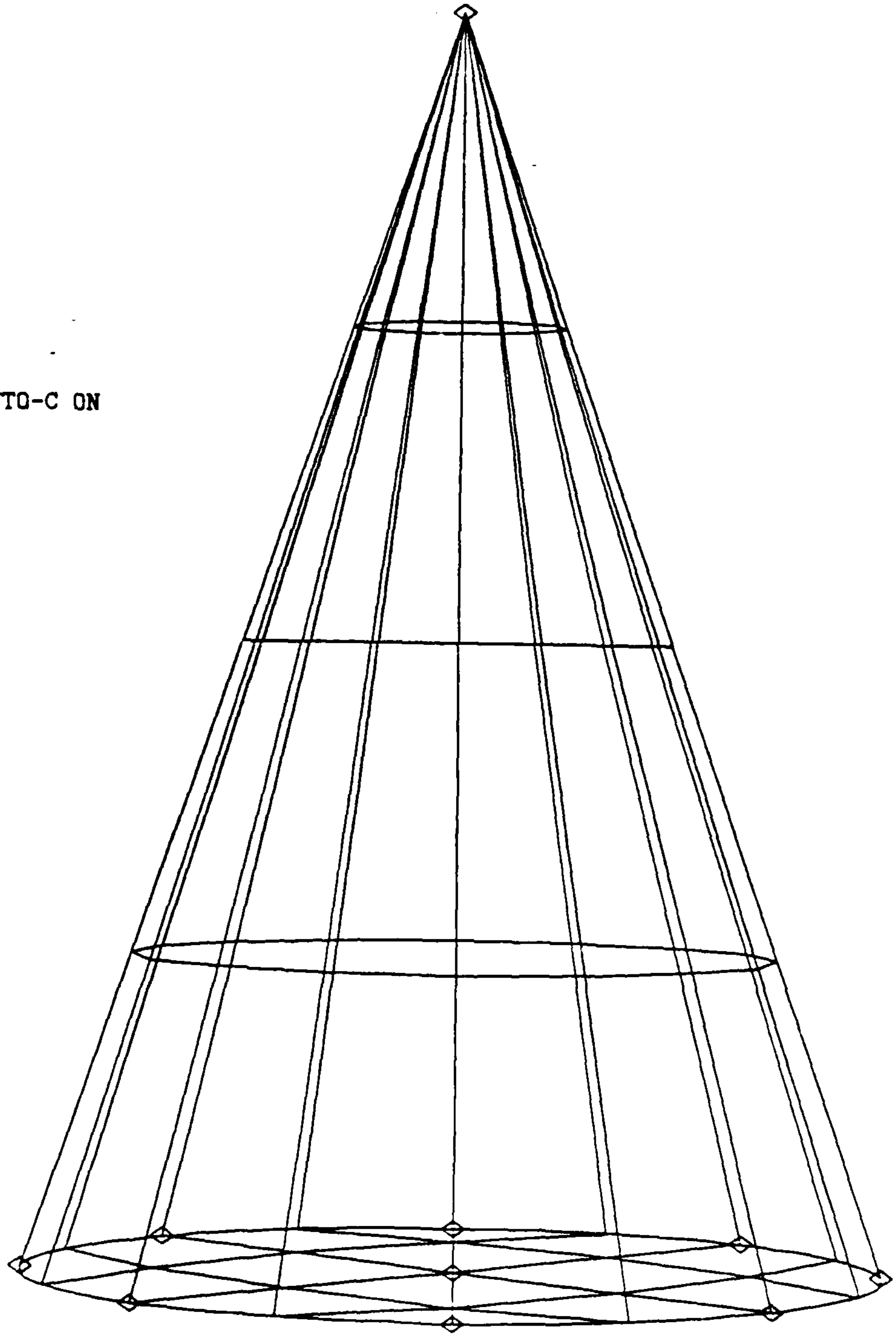
The third section contains an example with CALIPER in the physical fit compatibility problem. In this example, the user is given models of an F-15 wing, pylon, runway and an MK84 weapon. It is desired to place the weapon on the wing/pylon assembly so that there are no interferences.

The fourth section contains several optimal packaging examples. These examples take the user through model creation, meshing, mass properties calculation, design setup and optimal packaging with PACKER. There are many heuristic pointers and discussions in the right margin of the example outputs.

Finally, there is a general, useful, modeling guideline providing general rules of thumb and pointers for making good geometric models. A trouble shooting guide is also provided that suggests some proven 'work arounds'.

The following NAVGRAPH run file creates the cone shown.

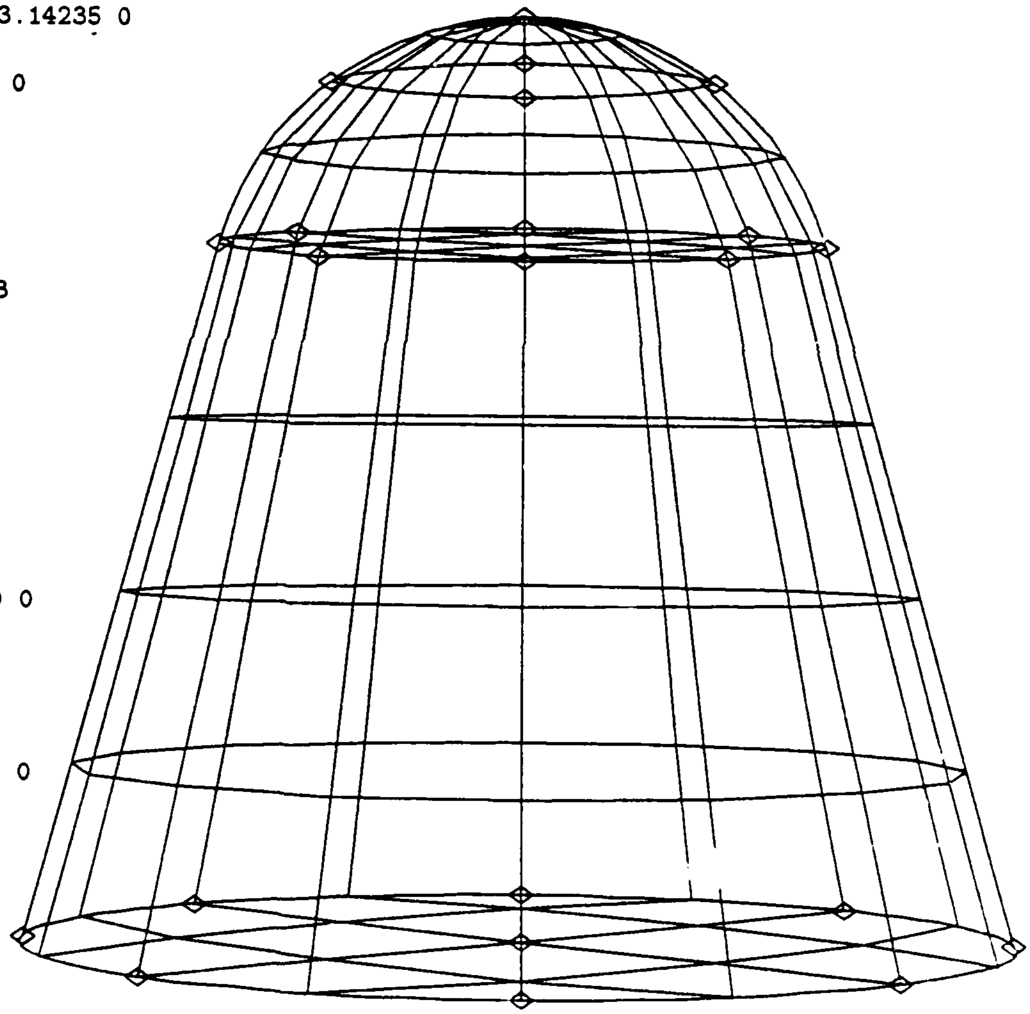
```
DEFIN
POINT
ENTE
1
0 0 0
2
6 2 0
3
6 0 0
0
DR
DISP AUTO-C ON
DR

LINE
ENTE
1
1 2
1 3

ROTA
3
1
270 0 0
3
DRAW
SWEE
6
2
2
90
ROTA
7
6
270 0 0
3
ENTE
10
1 1
```



```
DR
SOLID
LINE
1
9 8 7 6
10 10 10 10
1 5 4 3

DR
EXIT


Y
```

This file creates a cone with a nose cap. The model is shown.

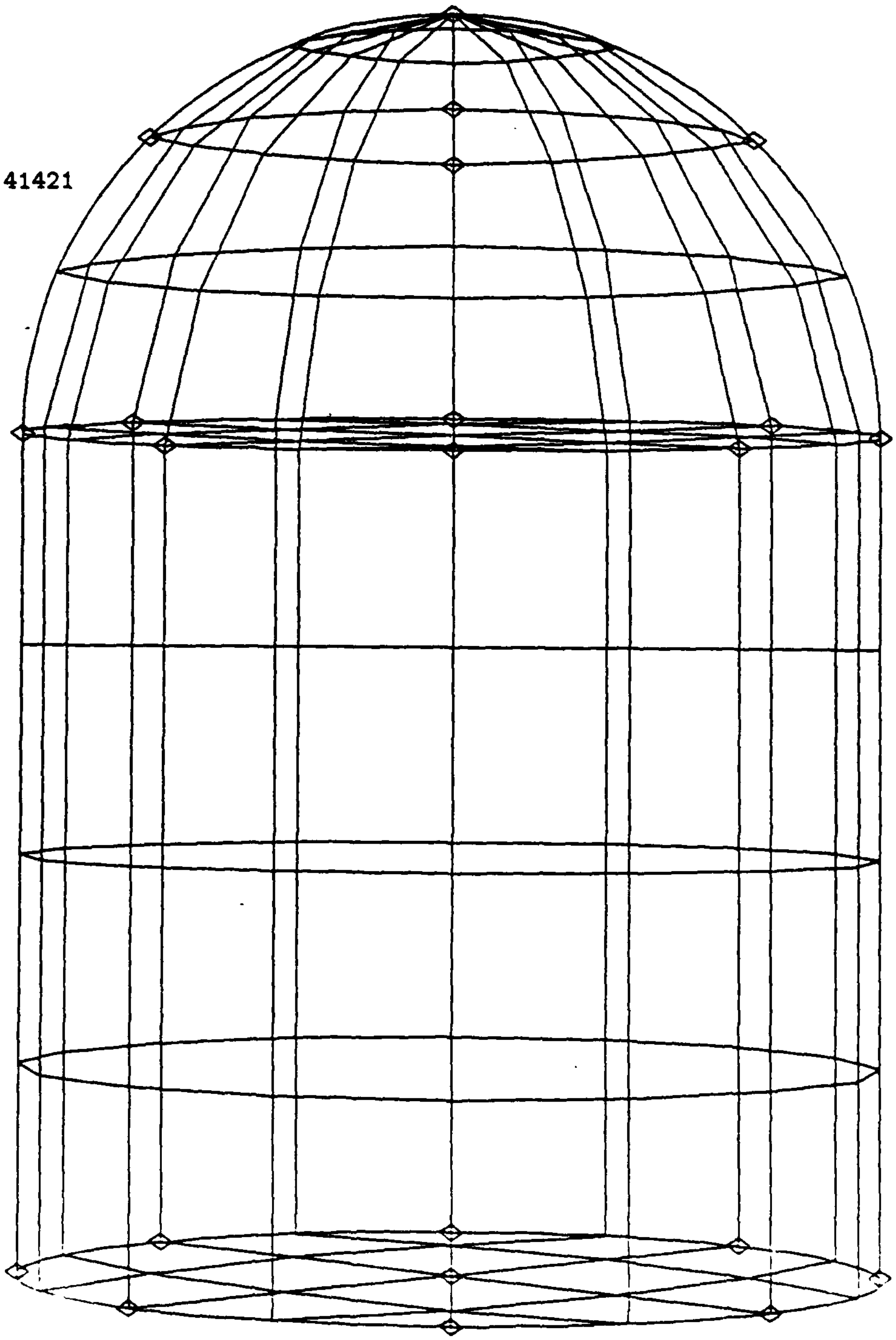
```
DEFI
POIN
ENTE
1
0 0 0
2
.413293 1.217523 0
3
1.482362 1.931852 0
4
6.0 3.14235 0
5
6. 0 0
0

LINE
ARC
1
1 2 3
ENTE
2
3 4
1 5

ROTA
4
1
270 0 0
3
ROTA
7
2
270 0 0
3
SWEE
10
3
3
90
ROTA
11
```



```
10
270 0 0
3
SWEE
14
4
3
90
ROTA
15
14
270 0 0
3
ENTE
18
1 1


SOLI
LINE
1
17 16 15 14
13 12 11 10
2 9 8 7
LINE
2
13 12 11 10
18 18 18 18
1 6 5 4
```

This file creates a cylinder with a nose cap. The model is shown here.

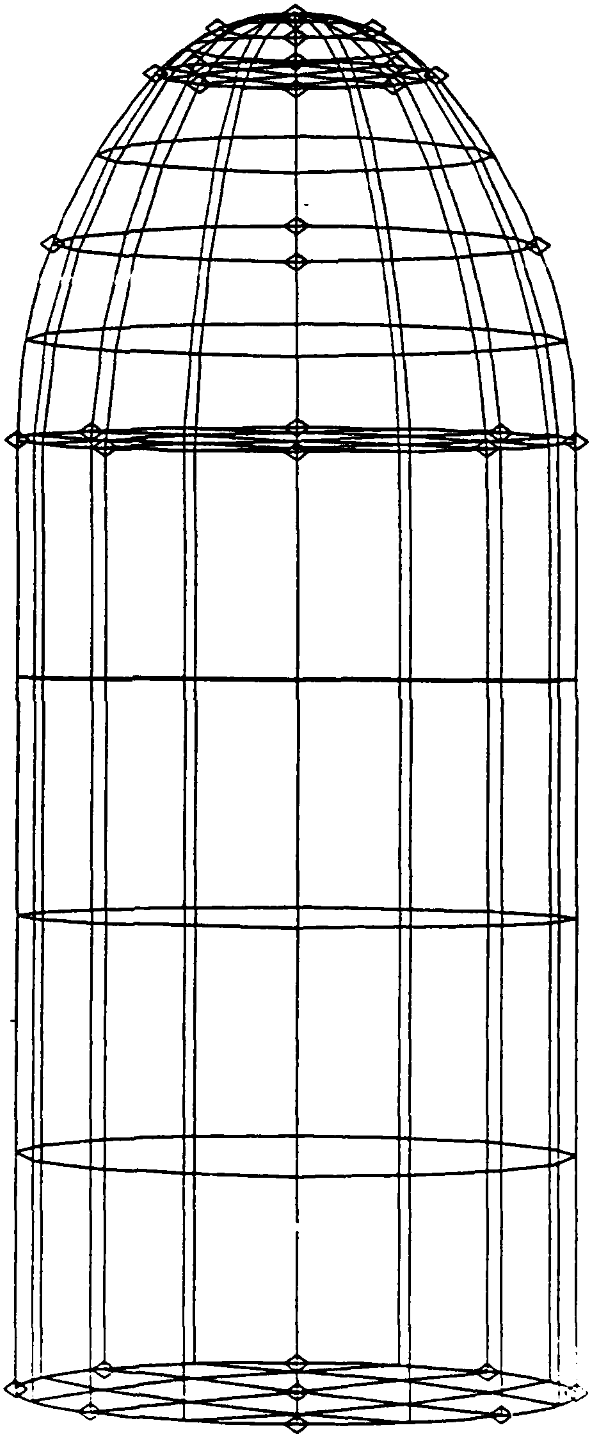
```
DEFI
POIN
ENTE
1
0 0 0
2
.58579 1.41421
3
2. 2. 0
4
6.0 2. 0
5
6. 0 0
0

LINE
ARC
1
1 2 3
ENTE
2
3 4
1 5

ROTA
4
1
270 0 0
3
ROTA
7
2
270 0 0
3
SWEE
10
3
3
90
ROTA
11
```



```
10
270 0 0
3
SWEE
14
4
3
90
ROTA
15
14
270 0 0
3
ENTE
18
1 1


SOLI
LINE
1
17 16 15 14
13 12 11 10
2 9 8 7
LINE
2
13 12 11 10
18 18 18 18
1 6 5 4
```

This file creates a tangent ogive body of revolution. The model is shown.

```
DEFI
POIN
ENTE
1
0 0 0
2
.177124 .82288 0
3
.67712 1.5 0
4
2.524431 2.612486 0
5
4.64575 3. 0
6
15. 3. 0
0

LINE
ARC
1
1 2 3
ARC
2
3 4 5
ENTE
3
5 6

ROTA
4
1
270 0 0
3
ROTA
7
2
270 0 0
3
ROTA
10
3
```



```
270 0 0
3

POIN
ENTE
31
15. 0 0
0

LINE
ENTE
13
1 31

SWEE
14
3
13
90
ROTA
15
14
270 0 0
3
SWEE
18
5
13
90
ROTA
19
18
270 0 0
3
SWEE
22
6
13
90
ROTA
23
22
270 0 0
```

```
3
ENTE
26
1 1


SOLI
LINE
1
25 24 23 22
21 20 19 18
3 12 11 10
LINE
2
21 20 19 18
17 16 15 14
2 9 8 7
LINE
3
17 16 15 14
26 26 26 26
1 6 5 4
```

The following are the MOVIESTAR commands for creating a solid model of a MAVERICK missile.

```
DISP AUTO-CEN ON                Turn on auto-center.
DEFI
POIN
ENTE
1
0 0 0
2
2.5 0 4
3                               Points for first solid.
2.5 2.8284 2.8284
4
2.5 4 0
5
.6764 0 2.3585
0

LINE
ENTE
1
1 1
1 1
1 1
1 1

ARC
5
2 3 4
ROTA                            Lines for first solid.
6
5
270 0 0
3
ARC
9
1 5 2
ROTA
10
9
270 0 0
3
```

```
SOLI
LINE
1
1 2 3 4                         First solid created from
5 6 7 8                         twelve lines.
9 10 11 12

POIN
ENTE
24
14 0 6
25
14 4.2426 4.2426                Points for second solid.
26
14 6 0
27
8.1637 0 5.4963
0

LINE
ARC
13
24 25 26
ROTA
14
13
270 0 0
3                               Lines for second solid.
ARC
17
2 27 24
ROTA
18
17
270 0 0
3

SOLI
LINE
2
5 6 7 8                         Second solid made with
13 14 15 16                     twelve lines.
```

```
17 18 19 20

SURF
TRAN
13                         Translate surface to be
12                         used in third solid.
83 0 0

SOLI
PARA                       Third solid made by parametric
3                          fit between two surfaces.
12 13

POIN
ENTE
54
31.5 -.5 6
55
31.5 .5 6
56
84.16 -.5 6                Points for large fin.
57
84.16 .5 6
58
84.16 .1 14.25
59
84.16 -.1 14.25
0

SOLI
ENTE
4                          Fin solid from points.
54 55 55 54
56 57 58 59

ROTA
5
4                          Rotate fin solid to create
270 0 0                    the other three large fins.
3

SURF
TRAN
```

42 23 4.63 0 0 TRAN 43 42 6 0 0	Translate surfaces to be used for small fins.
SOLI PARA 8 42 43 ROTA 9 8 270 0 0 3	Create small fin with a parametric fit between two surfaces. Rotate solid to create other fins.
GLOB GROU NEW ADD SOLI 1 2 3 4 5 6 7 8 9 10 11	Place all solids into a group.
SAVE MAV 1 0 1	
DISP ENT ALL OFF DISP ENT POST GALL ON DISP ENT GROU ON DRAW EXIT Y MAV.DAT Y N	Set display options to display groups only.

The following commands and procedure may be used to create and evaluate a physical-fit compatability of a weapon on an aircraft.

Begin by executing DBMERGE to get the following groups from their original files to create the file tcase1.dat.

GROUP	FILE
tc1wing	wing.geo
sweep	wingswp.geo
leinbpyl	inbpylon.geo
runway	runway.geo
mk84	mk84.geo

Now execute MOVIESTAR, open the file tcase1.dat and execute the following commands.

```
DISP ENT ALL OFF
DISP ENT GROU ON
GLOB COLO
GROU                          Set display-options to
TC1WING                       display only groups and
1 0 0                         give each group a different
GROU                          color so they will be easy
SWEEP                         to see.
0 1 0
GROU
LEINBPYL
1 1 0
GROU
MK84
0 0 1
GROU
RUNWAY
1 1 1

VIEW
ROT                           Orient the model to a
SCREE                         side-on configuration.
-90 0 0
```

DRAW	
GLOB	
GROU	
ROTA	Orient the weapon so it can
MK84	be placed on the pylon.
MK84	
45 0 180	
LIST	
GROU	
MK84	
GROU	
LENINBPYL	Get hook point numbers.
GROU	
LEINBPYL	
GROU	
SNAP	
LEINBPYL	After weapon is properly
6088	oriented we can use the
MK84	group snap command to place
6255	it in the proper position.
DRAW	
TRANS	
MK84	Now we'll translate it up
MK84	so it interferes with the
0 0 8	pylon.
NEW	
ADD	
SOLI	Create a new group of one of
53	the weapon solids for quick
	interference calculation.
SAVE	
MID	
0 0 1	
DRAW	

CALIPER	Execute caliper.
MID LEINBPYL Y 0 0 1	
HIGH 1 0 1 Y	Display interference highlight.
EXIT	
TCASE1.INT Y N	

Optimal packaging examples with PACKER.

Several different simple examples are given here of packaging objects to user defined objective functions and constraints.

The problems given here have only a few objects and a few container walls for simplicity. More complex problems have been solved and documented. These problems here are for demonstration and tutorial and do not reflect the type of complex problems that can be solved.

The first example is one where there is a single container wall and two objects and we wish to minimize the x centroid of the composite model. We setup to only translate, and put constraints on the problem so that there will be a separation between the objects and between the objects and the container wall of .1 inches. The example shows the input (and some output) through the complete problem.

The second example takes the model from the first and simply changes the objective function to minimize I_{xx} with an equality constraint on the x centroid (= 1.2). The solution shows how PACKER can find a feasible solution when the starting design is not feasible.

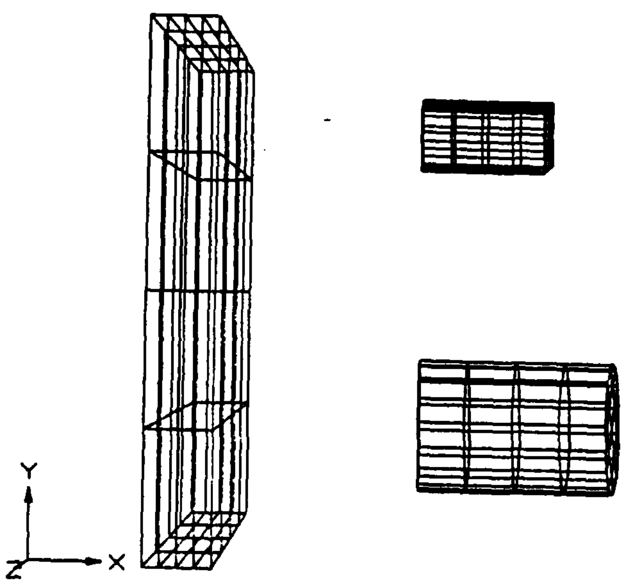
The last example is a simple one with rotational degrees of freedom. Here an object (cylinder) is oriented arbitrarily at the starting design. The objective is to minimize I_{xx}, only now the rotations will be found that orient the object in such a way that the object's axis of symmetry is lined up with the global axis (the optimal solution for a cylinder).

We begin with a NAVGRAPH session. The following command create the desired model shown.

```
DEFI
POIN
ENTE
1
0 -4 -4
2
0 4 -4
3
0 4 4
4
-4 4
4
0 -4 4
0
DR

SURF
ENTE
1
1 2 3 4

TRAN
2
1
1 0 0

SOLI
PARA                                  the container wall.
1
1 2

DISP AUTO-C ON
DR
POIN
ENTE
9
4 1 0
0
ROTA
10
9
315
```

```
7

LINE
ARC
13
9 10 11
ARC
14
11 12 13
ARC
15
15 14 13
ARC
16
9 16 15

SURF
LINE
7
13 14 15 16
TRAN
8
7
3 0 0

SOLI
PARA
2
7 8



DEFI
POIN
ENTE
25
4 2
26
6 2
27
6 3
28
4 3
0

SURF
```

a cylinder object.



```
ENTE
13
25 26 27 28

TRAN
14
13
0 0 2

SOLI
PARA
3                               a box.
13 14


GLOB
GROU
NEW
ADD
SOLI
1

SAVE
CONT                            define the container group.
1
NEW
ADD
SOL
2

SAVE
CYL1
0 1                             define the cylinder group.
NEW
ADD
SOLI
3

SAVE
BOX1                            define the bcx group.
0 0 1
DR

VIEW ROTA SCREE 0 -90
DRAW
```

GROUP	
TRAN	
CYL1	translate the cylinder.
CYL1	
0 -2 2	
DRAW	
GENE	
CREA	
ELEM	
SOLI	mesh the container with one
1	hexahedron finite element for
	mass properties calculation.
1 1 1	
S1	
SOLI	
2	mesh the cylinder.
10 10 1	10 elements along the curved
	sections of the cylinder.
S2	
SOLI	
3	mesh the box.
1 1 1	one hexahedron is enough for a box.
S3	
DRAW	
EXIT	

The model is created and saved, we call it PACK.DAT. Now we run MASSPROP and calculate the mass properties for each object. We need the volume, the density, the center of gravity and the local (centroidal) moments of inertial for each object. The output files from MASSPROP for each group is shown here. The arrow shows the information needed for input into the filename.MAS (PACK.MAS in this case).

```
NAVGRAPH DATA FILE = PACK
GROUP NAME          = CONT
# OF HEXAHEDRON ELEMENTS   =          1
# OF WEDGE ELEMENTS        =          0
# OF PENTAHEDRON ELEMENTS =           0
# OF TETRAHEDRON ELEMENTS =           0

SPECIFIC WEIGHT:     490.0000
GRAVITY CONST. :     32.20000
DENSITY        :     15.21739          <
TOTAL VOLUME   :     64.00000          <
TOTAL MASS     :     973.9130
GLOBAL XBAR    :    0.5000000          <
GLOBAL YBAR    :    6.2670026E-08      <
GLOBAL ZBAR    :    9.4005038E-08      <
GLOBAL IXX     :     10388.41
GLOBAL IYY     :     5518.840
GLOBAL IZZ     :     5518.840
GLOBAL IXY     :   -3.0517578E-05
GLOBAL IXZ     :   -3.0517578E-05
GLOBAL IYZ     :   -6.1035156E-05
CENTROID IXX   :     10388.41          <
CENTROID IYY   :     5275.362          <
CENTROID IZZ   :     5275.362          <
CENTROID IXY   :   -3.6379788E-12      <
CENTROID IXZ   :    1.5258785E-05      <
CENTROID IYZ   :   -6.1035149E-05      <


NAVGRAPH DATA FILE = PACK
GROUP NAME          = CYL1
# OF HEXAHEDRON ELEMENTS   =        100
# OF WEDGE ELEMENTS        =          0
# OF PENTAHEDRON ELEMENTS =           0
# OF TETRAHEDRON ELEMENTS =           0

SPECIFIC WEIGHT:     490.0000
GRAVITY CONST. :     32.20000
DENSITY        :     15.21739          <
```

```
TOTAL VOLUME     :     9.386064          <
TOTAL MASS       :     142.8314
GLOBAL XBAR      :     5.500000          <
GLOBAL YBAR      :    -2.000000          <
GLOBAL ZBAR      :     2.000000          <
GLOBAL IXX       :     1213.773
GLOBAL IYY       :     5034.661
GLOBAL IZZ       :     5034.661
GLOBAL IXY       :     1571.146
GLOBAL IXZ       :    -1571.145
GLOBAL IYZ       :     571.3256
CENTROID IXX     :     71.12195          <
CENTROID IYY     :     142.6846          <
CENTROID IZZ     :     142.6846          <
CENTROID IXY     :    0.0000000E+00      <
CENTROID IXZ     :    8.5449219E-04      <
CENTROID IYZ     :   -1.8310547E-04      <


NAVGRAPH DATA FILE = PACK
GROUP NAME         = BOX1
# OF HEXAHEDRON ELEMENTS  =          1
# OF WEDGE ELEMENTS       =          0
# OF PENTAHEDRON ELEMENTS =          0
# OF TETRAHEDRON ELEMENTS =          0

SPECIFIC WEIGHT:     490.0000
GRAVITY CONST. :     32.20000
DENSITY        :     15.21739            <
TOTAL VOLUME   :     4.000000            <
TOTAL MASS     :     60.86956
GLOBAL XBAR    :     5.000000            <
GLOBAL YBAR    :     2.500000            <
GLOBAL ZBAR    :    0.9999999            <
GLOBAL IXX     :     466.6667
GLOBAL IYY     :     1623.188
GLOBAL IZZ     :     1927.536
GLOBAL IXY     :    -760.8696
GLOBAL IXZ     :    -304.3478
GLOBAL IYZ     :    -152.1739
CENTROID IXX   :     25.36227            <
CENTROID IYY   :     40.57971            <
CENTROID IZZ   :     25.36230            <
CENTROID IXY   :    0.0000000E+00        <
CENTROID IXZ   :   -3.0517578E-05        <
CENTROID IYZ   :    0.0000000E+00        <
```

These data are put into the file PACK.MAS in free format in the following manner. Each object has two lines, the first line has the volume, density, x centroid, y centroid and z centroid. The second line has the six moments of inertia in the order above. The PACK.MAS file is shown here (notice the values from above).

```
64.0 15.21739 .5 0. 0.
10388.41 5275.362 5275.362 0. 0. 0.
9.386064 15.21739 5.5 -2.0 2.0
71.12195 142.6846 142.6846 0. 0. 0.
4.0 15.21739 5. 2.5 1.0
25.36227 40.57971 25.3623 0. 0. 0.
```

We are now ready to run PREPAC. PREPAC calculates all of the initial analsis functions from the initial analysis variables. The user is asked for the NAVGRAPH file name, the container name, the mass properties file name and a location for the center of pressure (optional). The user is also asked what the OPTDES problem file name is. In this problem we choose PACK.PRO.

SETUP is the next step. Here we define the design variables (tranlations and rotations) as a subset of the analysis variables and the design function (the objective and the constraints) as a subset of the analysis functions (separation distances and mass properties).

The following is what is seen on the screen in the SETUP session.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 SETUP PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> PACK.PRO             input the problem name

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
```

```
DIS
DISPLAY MENU:AV,AF,HELP
<SELECT> AV
AV#    VALUE      DESCRIPTION
---    -----      -----------
  1  0.00000E+00 1QUAT, GROUP CYL1          display analysis
  2  0.00000E+00 2QUAT, GROUP CYL1          variables
  3  0.00000E+00 3QUAT, GROUP CYL1
  4  0.00000E+00 1TRAN, GROUP CYL1
  5  0.00000E+00 2TRAN, GROUP CYL1
  6  0.00000E+00 3TRAN, GROUP CYL1
  7  0.00000E+00 1QUAT, GROUP BOX1
  8  0.00000E+00 2QUAT, GROUP BOX1
  9  0.00000E+00 3QUAT, GROUP BOX1
 10  0.00000E+00 1TRAN, GROUP BOX1
 11  0.00000E+00 2TRAN, GROUP BOX1
 12  0.00000E+00 3TRAN, GROUP BOX1

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> VAR                                 edit design variables

EDIT DESIGN VARIABLES
       (1) (2)     (3)            (4)  (5)          (6)
    DV# #MP AV#    COEFF     VALUE    MINIMUM     MAXIMUM  DESCRIPTION
    --- --- ---    -----     -----    -------     -------  -----------
  1)

<EDIT: C,G,J,Q,HELP> G 1 3 2 4 1 G 4 6 2 10 1
                                   select design variables as
                                   all tranlations.  Note, that
                                   by not defining any rotations
                                   the objects will not rotate.
EDIT DESIGN VARIABLES
       (1) (2)     (3)            (4)  (5)          (6)
    DV# #MP AV#    COEFF     VALUE    MINIMUM     MAXIMUM  DESCRIPTION
    --- --- ---    -----     -----    -------     -------  -----------
  1)  1   1   4    1.00     0.00E+00P -1.00        1.00    1TRAN, GROUP CYL1
  2)  2   1   5    1.00     0.00E+00P -1.00        1.00    2TRAN, GROUP CYL1
  3)  3   1   6    1.00     0.00E+00P -1.00        1.00    3TRAN, GROUP CYL1
  4)  4   1  10    1.00     0.00E+00P -1.00        1.00    1TRAN, GROUP BOX1
  5)  5   1  11    1.00     0.00E+00P -1.00        1.00    2TRAN, GROUP BOX1
  6)  6   1  12    1.00     0.00E+00P -1.00        1.00    3TRAN, GROUP BOX1
  7)

<EDIT: C,G,J,Q,HELP> G 1 6 5 -5 G 1 6 6 5
                                              edit how far the objects
```

```
                                                       can translate (min/max)
EDIT DESIGN VARIABLES
       (1) (2)    (3)               (4)  (5)          (6)
    DV# #MP AV#   COEFF      VALUE      MINIMUM    MAXIMUM  DESCRIPTION
    --- --- ---   -----      -----      -------    -------  -----------
  1)  1   1   4   1.00      0.00E+00P  -5.00        5.00    1TRAN, GROUP CYL1
  2)  2   1   5   1.00      0.00E+00P  -5.00        5.00    2TRAN, GROUP CYL1
  3)  3   1   6   1.00      0.00E+00P  -5.00        5.00    3TRAN, GROUP CYL1
  4)  4   1  10   1.00      0.00E+00P  -5.00        5.00    1TRAN, GROUP BOX1
  5)  5   1  11   1.00      0.00E+00P  -5.00        5.00    2TRAN, GROUP BOX1
  6)  6   1  12   1.00      0.00E+00P  -5.00        5.00    3TRAN, GROUP BOX1
  7)

<EDIT: C,G,J,Q,HELP> Q

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> DIS AF FUN
AF#    VALUE      DESCRIPTION
---    -----      -----------
  1   3.0000       1CON   SEP & CYL1
  2   3.0000       1CON   SEP & BOX1
  3   3.0000      CYL1  0-0 SEP BOX1
  4   5.7446       1CON   CEN & CYL1
  5   5.2440       1CON   CEN & BOX1
  6   4.6368      CYL1  0-0 CEN BOX1
  7   77.386      TOTAL VOLUME
  8   1177.6      TOTAL MASS                     display analysis
  9   1.3390       X MASS CENTROID               functions
 10 -0.11336       Y MASS CENTROID
 11  0.29427       Z MASS CENTROID
 12   11952.      XX MOMENT OF INERTIA
 13   9963.2      YY MOMENT OF INERTIA
 14   10354.      ZZ MOMENT OF INERTIA
 15   631.53      XY MOMENT OF INERTIA
 16  -1411.5      XZ MOMENT OF INERTIA
 17   379.87      YZ MOMENT OF INERTIA
 18   4.0000       X PRESSURE CENTER
 19  0.00000E+00   Y PRESSURE CENTER
 20  0.00000E+00   Z PRESSURE CENTER

EDIT DESIGN FUNCTIONS
       (1) (2)    (3)               (4)      (5)       (6)
    DF# #MP AF#   COEFF      VALUE      ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---   -----      -----      ---------   -----   -----------
  1)
```

```
<EDIT: C,G,J,Q,HELP> C 1 2 9 C 1 4 v        max xbar
                                             as objective
EDIT DESIGN FUNCTIONS
        (1) (2)     (3)               (4)    (5)         (6)
    DF# #MP AF#    COEFF      VALUE       ALLOWABLE     RANGE    DESCRIPTION
    --- --- ---    -----      -----       ---------     -----    -----------
  1)  1   1   9    1.00        1.3     vP 0.000E+00     1.00      X MASS CENTROID
  2)

<EDIT: C,G,J,Q,HELP> G 2 4 2 1 1             select contraints
                                             as separations

EDIT DESIGN FUNCTIONS
        (1) (2)     (3)               (4)    (5)         (6)
    DF# #MP-AF#    COEFF      VALUE       ALLOWABLE     RANGE    DESCRIPTION
    --- --- ---    -----      -----       ---------     -----    -----------
  1)  1   1   9    1.00        1.3     vP 0.000E+00     1.00      X MASS CENTROID
  2)  2   1   1    1.00        3.0     <P 0.000E+00     1.00      1CON  SEP & CYL1
  3)  3   1   2    1.00        3.0     <P 0.000E+00     1.00      1CON  SEP & BOX1
  4)  4   1   3    1.00        3.0     <P 0.000E+00     1.00     CYL1  0-0 SEP BOX1
  5)

<EDIT: C,G,J,Q,HELP> G 2 4 4 > G 2 4 5 .1    make them inequality
                                             greater than .1

EDIT DESIGN FUNCTIONS
        (1) (2)     (3)               (4)    (5)         (6)
    DF# #MP AF#    COEFF      VALUE       ALLOWABLE     RANGE    DESCRIPTION
    --- --- ---    -----      -----       ---------     -----    -----------
  1)  1   1   9    1.00        1.3     vP 0.000E+00     1.00      X MASS CENTROID
  2)  2   1   1    1.00        3.0     >P 0.100         1.00      1CON  SEP & CYL1
  3)  3   1   2    1.00        3.0     >P 0.100         1.00      1CON  SEP & BOX1
  4)  4   1   3    1.00        3.0     >P 0.100         1.00     CYL1  0-0 SEP BOX1
  5)

<EDIT: C,G,J,Q,HELP> Q

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> QUI                                 we are ready to
                                             optimize with PACKER

<STORE THIS PROBLEM?> Y

<NAME OF PROBLEM FILE> PACK.PRO
```

We can now run PACKER for the optimization session.

The following is what is seen on the screen during the PACKER session.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 DESIGN PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> PACK.PRO

**** A PACKER BANNER IS SHOWN HERE****
The user is asked for the NAVGRAPH file, the container name, the
mass properties file, and the center of pressure (optional) again.

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRG

EDIT GENERALIZED REDUCED GRADIENT PARAMETERS
              VALUE           DESCRIPTION
              -----           -----------
( 1)            0             ALGORITHM PRINTOUT (1=ON 0=OFF)
( 2)      0.100000E-02        CONVERGENCE TOLERANCE
( 3)      0.100000E-04        MINIMUM STEPSIZE
( 4)      0.200000E-01        INITIAL STEPSIZE
( 5)      0.100000            MAX BOUND VIOLATION
( 6)      0.100000E-02        CONSTRAINT TOLERANCE

<EDIT: C,G,J,Q,HELP> C 1 1 Q        edit to have GRG print
                                    out more info to the screen

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRA
```

```
EDIT GRADIENT PARAMETERS
                VALUE              DESCRIPTION
                -----              -----------
( 1)              1                GRADIENT TYPE (1=FINITE DIFF,2=EXPLICIT)
( 2)        0.100000E-03           SCALED PERTURBATION FOR F-D GRADIENTS

<EDIT: C,G,J,Q,HELP> C 1 2 Q               edit for explicit
                                           gradients to be on

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PRO POS                           post process, this gives a
                                           NAVGRAPH menu for graphics.
```

```
DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> RUN GRG 3                                 run 3 GRG iterations
OBJECTIVE FUNCTION GRADIENT
 0.60644      0.00000E+00 0.00000E+00 0.25844      0.00000E+00 0.00000E+00

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00

LENGTH OF SEARCH DIRECTION VECTOR=    0.65173
SEARCH DIRECTION
-0.88994      0.13808E-05-0.17320E-05-0.45607      0.00000E+00 0.00000E+00

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
```

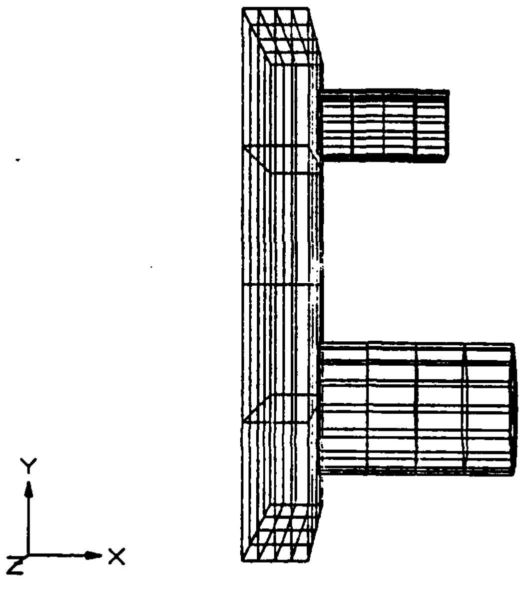
```
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01  optimization
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01  search.
LINE SEARCH STEP  6 STEP LENGTH= 5.7000E-01
LINE SEARCH STEP  7 STEP LENGTH= 8.2000E-01
CONSTRAINT   1 NOT FORMERLY BINDING
NEWTON-RAPHSON ITERATION
CONVERGENCE IN NEWTON-RAPHSON

DESIGN NUMBER     1
   SCALED STEP LENGTH              =  0.6517281
   SCALED OBJECTIVE       (DF#  1) =  0.9104874
   SCALED MAX INEQUALITY (DF#  2) =  0.1025966E-05
   ANALYSIS CALLS =     9     GRADIENT CALLS =     1
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.60644     0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00

LAGRANGIAN FUNCTION GRADIENT
 0.60644     0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00

LENGTH OF SEARCH DIRECTION VECTOR=    0.28276
SEARCH DIRECTION
 0.55326E-06 0.63876E-03 0.27091E-03 -1.0000     0.35697E-04 0.26065E-04

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02  again.
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01
CONSTRAINT   2 NOT FORMERLY BINDING
NEWTON-RAPHSON ITERATION
CONVERGENCE IN NEWTON-RAPHSON

DESIGN NUMBER     2
   SCALED STEP LENGTH              =  0.2827642
   SCALED OBJECTIVE       (DF#  1) =  0.8374085
   SCALED MAX INEQUALITY (DF#  3) =  0.3619690E-06
   ANALYSIS CALLS =    16     GRADIENT CALLS =     2
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.60644     0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00

LAGRANGIAN FUNCTION GRADIENT
 0.60644     0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00
```

```
LENGTH OF SEARCH DIRECTION VECTOR=     0.00377
SEARCH DIRECTION
-0.51617E-02 0.73796     0.64065    -0.55084E-03-0.57649E-01 0.20404

LINE SEARCH STEP  1 STEP LENGTH= 3.7690E-03
LINE SEARCH STEP  2 STEP LENGTH= 7.5380E-03
LINE SEARCH STEP  3 STEP LENGTH= 1.5076E-02  again.
LINE SEARCH STEP  4 STEP LENGTH= 3.0152E-02
LINE SEARCH STEP  5 STEP LENGTH= 6.0304E-02
NEWTON-RAPHSON ITERATION
CONVERGENCE IN NEWTON-RAPHSON
FITTING PARABOLA
LINE SEARCH STEP  6 STEP LENGTH= 3.3913E-02
FITTING PARABOLA
LINE SEARCH STEP  7 STEP LENGTH= 2.4549E-02
FITTING PARABOLA
LINE SEARCH STEP  8 STEP LENGTH= 2.8872E-02
FITTING PARABOLA
LINE SEARCH STEP  9 STEP LENGTH= 3.1080E-02

DESIGN NUMBER     3
   SCALED STEP LENGTH              =  0.3015201E-01
   SCALED OBJECTIVE      (DF#  1) =  0.8373099
   SCALED MAX INEQUALITY (DF#  2) =  0.7766739E-03
   ANALYSIS CALLS =    27     GRADIENT CALLS =     3


DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> RUN GRG 1                                  run one more GRG run.
OBJECTIVE FUNCTION GRADIENT
 0.60644      0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00

LAGRANGIAN FUNCTION GRADIENT
 0.60644      0.00000E+00 0.00000E+00 0.25844     0.00000E+00 0.00000E+00

LENGTH OF SEARCH DIRECTION VECTOR=     0.00524
SEARCH DIRECTION
 0.22105E-01 0.89388E-01 0.99486     0.31673E-02 0.11111E-01 0.40634E-01

LINE SEARCH STEP  1 STEP LENGTH= 5.2357E-03

DESIGN NUMBER     4
   SCALED STEP LENGTH              =  0.0000000E+00
   SCALED OBJECTIVE      (DF#  1) =  0.8373099
   SCALED MAX INEQUALITY (DF#  2) =  0.7766739E-03
```

ANALYSIS CALLS = 29 GRADIENT CALLS = 4

OPTIMUM ASSUMED (MINIMUM STEPSIZE REACHED) we reached an optimal solution

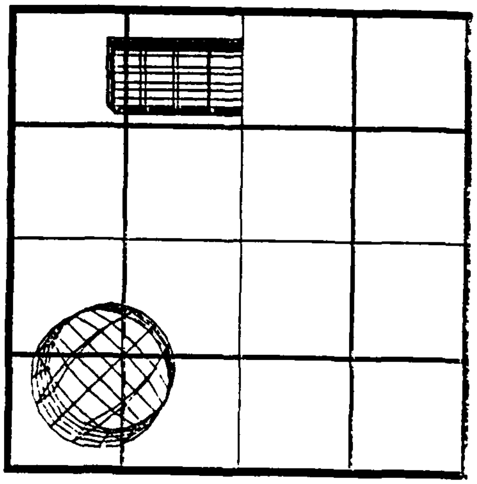
DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PRO POS draw it with NAVGRAPH to visually verify



DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> QUI

The second example starts with the same model as the first. We start with the same initial design, therefore, we do not need to remodel or recalculate the mass propeties. We simply run SETUP again and redefine the objective and contraints, and maybe add more design variables. In this example, we will minimize I_{xx} while contraining the global composite x centroid to be 1.2.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 SETUP PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> pack.pro                 start with the previous
                                                starting design.

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> dis af fun
AF#    VALUE      DESCRIPTION
---    -----      -----------
  1   3.0000       1CON  SEP & CYL1
  2   3.0000       1CON  SEP & BOX1
  3   3.0000      CYL1  O-O SEP BOX1
  4   5.7446       1CON  CEN & CYL1
  5   5.2440       1CON  CEN & BOX1
  6   4.6368      CYL1  O-O CEN BOX1
  7   77.386      TOTAL VOLUME
  8   1177.6      TOTAL MASS
  9   1.3390       X MASS CENTROID
 10 -0.11336       Y MASS CENTROID
 11  0.29427       Z MASS CENTROID
 12   11952.      XX MOMENT OF INERTIA
 13   9963.2      YY MOMENT OF INERTIA
 14   10354.      ZZ MOMENT OF INERTIA
 15   631.53      XY MOMENT OF INERTIA
```



```
 16  -1411.5      XZ MOMENT OF INERTIA
 17   379.87      YZ MOMENT OF INERTIA
 18   4.0000       X PRESSURE CENTER
 19  0.00000E+00  Y PRESSURE CENTER
 20  0.00000E+00  Z PRESSURE CENTER

EDIT DESIGN FUNCTIONS
        (1) (2)    (3)                (4)    (5)       (6)
    DF# #MP AF#    COEFF      VALUE       ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---    -----      -----       ---------   -----   -----------
  1)  1   1   9    1.00       1.3      vP 0.000E+00   1.00     X MASS CENTROID
  2)  2   1   1    1.00       3.0      >P 0.100       1.00     1CON   SEP & CYL1
  3)  3   1   2    1.00       3.0      >P 0.100       1.00     1CON   SEP & BOX1
  4)  4   1   3    1.00       3.0      >P 0.100       1.00    CYL1   0-0 SEP BOX1
  5)

<EDIT: C,G,J,Q,HELP> c 1 2 12                       change the objective
                                                    function to minimize Ixx

EDIT DESIGN FUNCTIONS
        (1) (2)    (3)                (4)    (5)       (6)
    DF# #MP AF#    COEFF      VALUE       ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---    -----      -----       ---------   -----   -----------
  1)  1   1  12    1.00     0.12E+05vP 0.000E+00   1.00    XX MOMENT OF INERTIA
  2)  2   1   1    1.00       3.0      >P 0.100       1.00     1CON   SEP & CYL1
  3)  3   1   2    1.00       3.0      >P 0.100       1.00     1CON   SEP & BOX1
  4)  4   1   3    1.00       3.0      >P 0.100       1.00    CYL1   0-0 SEP BOX1
  5)

<EDIT: C,G,J,Q,HELP> c 5 2 9 c 5 4 > c 5 5 1.2
                                                      add equality
                                                      constraint on
                                                      x centroid

EDIT DESIGN FUNCTIONS
        (1) (2)    (3)                (4)    (5)       (6)
    DF# #MP AF#    COEFF      VALUE       ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---    -----      -----       ---------   -----   -----------
  1)  1   1  12    1.00     0.12E+05vP 0.000E+00   1.00    XX MOMENT OF INERTIA
  2)  2   1   1    1.00       3.0      >P 0.100       1.00     1CON   SEP & CYL1
  3)  3   1   2    1.00       3.0      >P 0.100       1.00     1CON   SEP & BOX1
  4)  4   1   3    1.00       3.0      >P 0.100       1.00    CYL1   0-0 SEP BOX1
  5)  5   1   9    1.00       1.3      >P  1.20       1.00     X MASS CENTROID
  6)

<EDIT: C,G,J,Q,HELP> q
```

```
SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> qui

<STORE THIS PROBLEM?> y

<NAME OF PROBLEM FILE> pack2.pro                 call the problem file
                                                 something new.
```

Now for the optimization in PACKER.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 DESIGN PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> PACK2.PRO


****THE PACKER BANNER HERE***** and the user input PACK.DAT,
the container name, PACK.MAS again.


DESIGN NUMBER      0
   SCALED OBJECTIVE       (DF#  1) =   11951.75
   SCALED MAX INEQUALITY (DF#  4) =  -2.900000
   SCALED MAX EQUALITY    (DF#  5) =  0.1390441
   ANALYSIS CALLS =      0     GRADIENT CALLS =      0

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRG C 1 1 Q
```

```
DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRA C 1 2                      turn on explicit
                                            gradients again. this
                                            is important, because
                                            finite difference
                                            gradients can take a
                                            very long time.

EDIT GRADIENT PARAMETERS
              VALUE            DESCRIPTION
              -----            -----------
( 1)            2              GRADIENT TYPE (1=FINITE DIFF,2=EXPLICIT)
( 2)      0.100000E-03         SCALED PERTURBATION FOR F-D GRADIENTS

<EDIT: C,G,J,Q,HELP> Q

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> RUN GRG 5                          run 5 GRG iterations.

SEARCHING FOR FEASIBLE STARTING POINT       note that we started
                                            with a non-feasible
                                            design.  The equality
                                            contraint on the
                                            x centroid is violated.
                                            Therefore, it becomes
                                            the temporary objective.

TEMPORARY OBJECTIVE IS TO MINIMIZE DESIGN FUNCTION    5

OBJECTIVE FUNCTION GRADIENT
 0.16864     0.00000E+00 0.00000E+00 0.71870E-01 0.00000E+00 0.00000E+00

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00

LENGTH OF SEARCH DIRECTION VECTOR=    0.21092
SEARCH DIRECTION
-0.91995     0.10337E-03-0.12966E-03-0.39205    -0.50973E-04 0.96224E-04

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01
FITTING PARABOLA
LINE SEARCH STEP  6 STEP LENGTH= 2.1092E-01
```

```
FITTING PARABOLA

DESIGN NUMBER     1
   SCALED STEP LENGTH              =  0.2109229
   SCALED OBJECTIVE      (DF#  1) =   11951.61
   SCALED MAX INEQUALITY (DF#  2) =  -1.929813
   SCALED MAX EQUALITY   (DF#  5) =  0.0000000E+00
   ANALYSIS CALLS =     6     GRADIENT CALLS =     1

FEASIBLE POINT FOUND, BEGINNING OPTIMIZATION OF PROBLEM
                                                         we found
                                                         a feasible start
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -2694.6      2436.1      0.00000E+00  1590.7      429.65

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00

LENGTH OF SEARCH DIRECTION VECTOR=    1.78223
SEARCH DIRECTION
-0.20786E-04 0.56108    -0.56108     0.48774E-04 0.23567    -0.56111

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02      now we are
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01      working on
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01      minimizing
LINE SEARCH STEP  6 STEP LENGTH= 5.7000E-01      Ixx.
LINE SEARCH STEP  7 STEP LENGTH= 8.2000E-01
FITTING PARABOLA
LINE SEARCH STEP  8 STEP LENGTH= 5.8527E-01

DESIGN NUMBER     2
   SCALED STEP LENGTH              =  0.5852728
   SCALED OBJECTIVE      (DF#  1) =   11148.34
   SCALED MAX INEQUALITY (DF#  2) =  -1.929752
   SCALED MAX EQUALITY   (DF#  5) =  0.5551115E-16
   ANALYSIS CALLS =    14     GRADIENT CALLS =     2
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -684.75      496.63      0.00000E+00  1867.6     -396.95

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -684.75      496.63      0.00000E+00  1867.6     -396.95

LENGTH OF SEARCH DIRECTION VECTOR=    1.44810
```

```
SEARCH DIRECTION
-0.26263E-04-0.40811    -0.46377     0.61626E-04-0.67705     0.39995

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01
LINE SEARCH STEP  6 STEP LENGTH= 5.7000E-01
FITTING PARABOLA
LINE SEARCH STEP  7 STEP LENGTH= 3.3411E-01

DESIGN NUMBER     3
   SCALED STEP LENGTH              =  0.3341135
   SCALED OBJECTIVE      (DF#  1) =   10918.79
   SCALED MAX INEQUALITY (DF#  4) =  -1.498272
   SCALED MAX EQUALITY   (DF#  5) =  0.8326673E-16
   ANALYSIS CALLS =    21     GRADIENT CALLS =     3
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -1456.9     -525.08     0.00000E+00  1265.0      45.920

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -1456.9     -525.08     0.00000E+00  1265.0      45.920

LENGTH OF SEARCH DIRECTION VECTOR=    0.29450
SEARCH DIRECTION
 0.76464E-03 0.38627     0.61437    -0.17942E-02-0.63124     0.27364

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01
CONSTRAINT    3 NOT FORMERLY BINDING
CONVERGENCE IN NEWTON-RAPHSON

DESIGN NUMBER     4
   SCALED STEP LENGTH              =  0.2944839
   SCALED OBJECTIVE      (DF#  1) =   10631.58
   SCALED MAX INEQUALITY (DF#  4) = -0.6819350E-04
   SCALED MAX EQUALITY   (DF#  5) =  0.2775558E-16
   ANALYSIS CALLS =    27     GRADIENT CALLS =     4
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -674.48      580.53     0.00000E+00  686.51      211.71
```

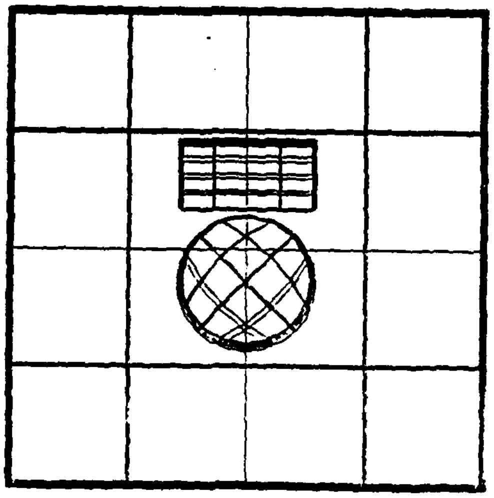
```
LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -674.48      580.53     0.00000E+00  686.51      211.71

LENGTH OF SEARCH DIRECTION VECTOR=    0.45280
SEARCH DIRECTION
-0.39206E-02-0.24511    -0.21417     0.91998E-02-0.24533    -0.91311

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
FITTING PARABOLA
LINE SEARCH STEP  5 STEP LENGTH= 1.0495E-01

DESIGN NUMBER      5                                 we have gone 5
   SCALED STEP LENGTH             = 0.1049469    iterations and are
   SCALED OBJECTIVE      (DF#  1) =  10614.75    not at the optimum.
   SCALED MAX INEQUALITY (DF#  4) = 0.4523932E-04
   SCALED MAX EQUALITY   (DF#  5) = 0.2775558E-16
   ANALYSIS CALLS =    32     GRADIENT CALLS =     5


DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> pro pos                                     we go to NAVGRAPH to
                                                     draw the current
                                                     configuration.

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> run grg 3                                   run 3 more iterations.
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -826.41      474.86     0.00000E+00  621.70     -56.573

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -674.48      580.53     0.00000E+00  686.51      211.71

LENGTH OF SEARCH DIRECTION VECTOR=    1.48955
SEARCH DIRECTION
-0.35937E-02 0.48331    -0.45321     0.84326E-02 0.48286     0.57251

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
FITTING PARABOLA
LINE SEARCH STEP  5 STEP LENGTH= 7.8947E-02
```

```
DESIGN NUMBER     6
   SCALED STEP LENGTH                 =  0.7894733E-01
   SCALED OBJECTIVE       (DF#  1) =   10601.06
   SCALED MAX INEQUALITY (DF#  4) =  0.2239870E-03
   SCALED MAX EQUALITY    (DF#  5) =  0.0000000E+00
   ANALYSIS CALLS =     37      GRADIENT CALLS =      6
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -601.03      233.64     0.00000E+00  717.64      87.084

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -601.03      233.64     0.00000E+00  717.64      87.084

LENGTH OF SÉARCH DIRECTION VECTOR=    1.01017
SEARCH DIRECTION
 0.22125E-02-0.17708    -0.63286    -0.51917E-02-0.17716    -0.73261

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02
FITTING PARABOLA
LINE SEARCH STEP  4 STEP LENGTH= 5.8321E-02

DESIGN NUMBER     7
   SCALED STEP LENGTH                 =  0.5832070E-01
   SCALED OBJECTIVE       (DF#  1) =   10594.28
   SCALED MAX INEQUALITY (DF#  4) =  0.2479473E-03
   SCALED MAX EQUALITY    (DF#  5) =  0.0000000E+00
   ANALYSIS CALLS =     41      GRADIENT CALLS =      7
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.00000E+00 -662.03      17.795     0.00000E+00  691.63      -22.605

LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 -662.03      17.795     0.00000E+00  691.63      -22.605

LENGTH OF SEARCH DIRECTION VECTOR=    0.53193
SEARCH DIRECTION
 0.56900E-02-0.46010    -0.18861    -0.13352E-01-0.45932     0.73590

LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 9.0121E-03

DESIGN NUMBER     8                                      still not there, but
   SCALED STEP LENGTH                 =  0.9012106E-02   very, very close.
```

```
SCALED OBJECTIVE      (DF#  1) =   10594.13       The step length is
SCALED MAX INEQUALITY (DF#  4) =  0.2119228E-03   small compared to the
SCALED MAX EQUALITY   (DF#  5) =  0.0000000E+00   first iteration.
ANALYSIS CALLS =    43     GRADIENT CALLS =     8
```

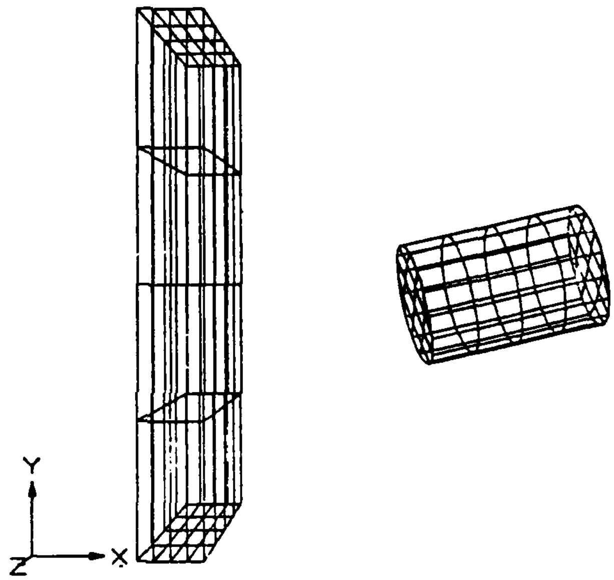
```
DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> pro pos                               go draw another display.
                                               we see that we are
                                               practically at the
                                               optimum.  This objective
                                               pulled the objects in
                                               toward the center.
```

```
DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> qui
```



In this last problem, we have a container wall and a cylinder. We will use rotational degrees of freedom in this problem. The model is created in NAVGRAPH with the following commands.

```
DEFI
POIN
ENTE
1
0 -4 -4
2
0 4 -4
3
0 4 4
4
-4 4
4
0 -4 4
0
DR

SURF
ENTE
1
1 2 3 4

TRAN
2
1
1 0 0

SOLI
PARA
1
1 2

DISP AUTO-C ON
DR
POIN
ENTE
9
4 1 0
0
ROTA
10
9
```



```
315
7

LINE
ARC
13
9 10 11
ARC
14
11 12 13
ARC
15
15 14 13
ARC
16
9 16 15

SUR
LINE
7
13 14 15 16
TRAN
8
7
3 0 0

SOLI
PARA
2
7 8



gene
crea
ele
sol
1



1 1 1
```

Create the finite elements for the mass properties calculation. Only those elements that are associated with the solids in the group are included in the mass properties calculation for the group.

```
s1


glo grou
new
add
sol
1

save
CONT
1
NEW
ADD
SOL
2

SAVE
CYL
0 1


QROTA                              Rotate the cylinder with
                                   the quaternion rotate command.

5.5 0 0
1 1 1
20
DR


GENE CREA ELE SOL                  mesh the cylinder



10 10 1



S2
```

We calculate the mass properties with MASSPROP. The cylinder's mass properties output file is listed here (the container is the same as in example one).

```
NAVGRAPH DATA FILE = PCYL
GROUP NAME          = CYL
# OF HEXAHEDRON ELEMENTS  =          100
# OF WEDGE ELEMENTS       =            0
# OF PENTAHEDRON ELEMENTS =            0
# OF TETRAHEDRON ELEMENTS =            0


SPECIFIC WEIGHT:     490.0000
GRAVITY CONST. :     32.20000
DENSITY        :     15.21739          <
TOTAL VOLUME   :     9.386071          <
TOTAL MASS     :     142.8315
GLOBAL XBAR    :     5.500001          <
GLOBAL YBAR    :    1.9488257E-07      <
GLOBAL ZBAR    :   -3.8444448E-08      <
GLOBAL IXX     :     76.76131
GLOBAL IYY     :     4459.951
GLOBAL IZZ     :     4461.088
GLOBAL IXY     :    -14.51823
GLOBAL IXZ     :     12.16859
GLOBAL IYZ     :     2.371527
CENTROID IXX   :     76.76131          <
CENTROID IYY   :     139.2964          <
CENTROID IZZ   :     140.4336          <
CENTROID IXY   :    -14.51807          <
CENTROID IXZ   :     12.16856          <
CENTROID IYZ   :     2.371527          <
```

The mass properties input file, PCYL.MAS, is shown here.

```
64.0 15.21739 .5 0. 0.
10388.41 5275.362 5275.362 0. 0. 0.
9.386064 15.21739 5.5 .0 .0
76.76131 139.2964 140.4336 -14.518 12.1686 2.37153
```

Now we run PREPAC. We give the NAVGRAPH file name PCYL.DAT, the container name, and PCYL.MAS. We call the problem file PCYL.PRO.

Once PREPAC has calculated the initial separation distance and the composite mass properties, we run SETUP.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 SETUP PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> PCYL.PRO

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
DIS AV

<SELECT> AV
AV#     VALUE      DESCRIPTION
---     -----      -----------
  1  0.00000E+00 1QUAT, GROUP CYL
  2  0.00000E+00 2QUAT, GROUP CYL
  3  0.00000E+00 3QUAT, GROUP CYL
  4  0.00000E+00 1TRAN, GROUP CYL
  5  0.00000E+00 2TRAN, GROUP CYL
  6  0.00000E+00 3TRAN, GROUP CYL

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> VAR

EDIT DESIGN VARIABLES
        (1) (2)    (3)            (4)  (5)        (6)
    DV# #MP AV#   COEFF     VALUE    MINIMUM    MAXIMUM  DESCRIPTION
    --- --- ---   -----     -----    -------    -------  -----------
  1)

<EDIT: C,G,J,Q,HELP> G 1 3 2 1 1      chose just the rotations

EDIT DESIGN VARIABLES
        (1) (2)    (3)            (4)  (5)        (6)
    DV# #MP AV#   COEFF     VALUE    MINIMUM    MAXIMUM  DESCRIPTION
```

```
  --- --- ---  -----       -----      -------     -------   -----------
1)  1   1   1   1.00       0.00E+00P -1.00        1.00     1QUAT, GROUP CYL
2)  2   1   2   1.00       0.00E+00P -1.00        1.00     2QUAT, GROUP CYL
3)  3   1   3   1.00       0.00E+00P -1.00        1.00     3QUAT, GROUP CYL
4)

<EDIT: C,G,J,Q,HELP> Q

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> DIS AF FUN
AF#   VALUE       DESCRIPTION
---   -----       -----------
  1   2.7796       1CON  SEP & CYL
  2   5.0000       1CON  CEN & CYL
  3   73.386      TOTAL VOLUME
  4   1116.7      TOTAL MASS
  5   1.1395       X MASS CENTROID
  6  0.00000E+00   Y MASS CENTROID
  7  0.00000E+00   Z MASS CENTROID
  8   10465.      XX MOMENT OF INERTIA
  9   8528.7      YY MOMENT OF INERTIA
 10   8529.9      ZZ MOMENT OF INERTIA
 11  -14.518      XY MOMENT OF INERTIA
 12   12.169      XZ MOMENT OF INERTIA
 13   2.3715      YZ MOMENT OF INERTIA
 14  0.00000E+00   X PRESSURE CENTER
 15  0.00000E+00   Y PRESSURE CENTER
 16  0.00000E+00   Z PRESSURE CENTER

EDIT DESIGN FUNCTIONS
        (1) (2)    (3)              (4)     (5)        (6)
    DF# #MP AF#   COEFF       VALUE      ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---   -----       -----      ---------   -----   -----------
 1)

<EDIT: C,G,J,Q,HELP> C 1 2 8 C 1 4 v
                                           just an objective function needed
                                           since there will be no
                                           interference.

EDIT DESIGN FUNCTIONS
        (1) (2)    (3)              (4)     (5)        (6)
    DF# #MP AF#   COEFF       VALUE      ALLOWABLE   RANGE   DESCRIPTION
    --- --- ---   -----       -----      ---------   -----   -----------
 1)  1   1   8    1.00      0.10E+05vP 0.000E+00    1.00    XX MOMENT OF INERTIA
 2)
```

```
<EDIT: C,G,J,Q,HELP> Q

SETUP MENU:DIS,VAR,FUN,COM,QUI,HELP
<SELECT> QUI

<STORE THIS PROBLEM?> Y

<NAME OF PROBLEM FILE> PCYL.PRO
```

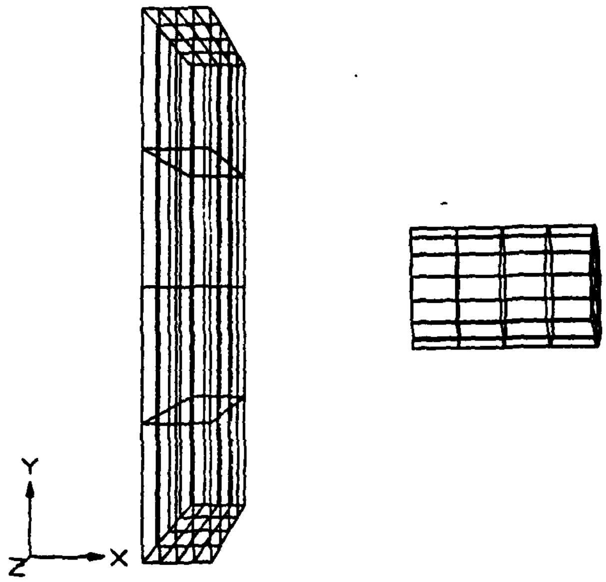
We now run PACKER.

```
WELCOME TO THE OPTDES.BYU VERSION 4.0 DESIGN PROGRAM
COPYRIGHT (C) 1988 BY BRIGHAM YOUNG UNIVERSITY

PROMPT TYPES:
   "<...>" TYPE RESPONSE
   "{...}" PRESS RETURN KEY

SPECIAL CHARACTERS:
   "*" ESCAPE TO MAIN MENU
   "%" EXECUTE OPERATING SYSTEM COMMAND
   "#" OPEN AND CLOSE INTERACTION FILES

INTERACTION FILE NAMES:
   "LOG" STARTING LOG FILE NAME
   "RES" STARTING RESPONSE FILE NAME
   "NUL" THERE IS NO STARTING BATCH FILE

<NAME OF PROBLEM FILE> PCYL.PRO

****The PACKER BANNER***** (not shown here to save space)
Input the NAVGRAPH file (PCYL.DAT), the container name, and the mass properties
file (PCYL.MAS).  The initial design looked like this:
```

```
DESIGN NUMBER     0
   SCALED OBJECTIVE      (DF#  1) =   10465.17
   ANALYSIS CALLS =     0     GRADIENT CALLS =     0

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRG C 1 1 Q

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PAR GRA C 1 2 Q                turn on explicit gradients

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> DIS DV DIS DF                  display design variables

DV#  SCALED VALUE  UNSCALED VALUE SCALED MIN SCALED MAX DESCRIPTION
---  ------------  -------------- ---------- ---------- -----------
  1  0.0000000E+00  0.0000000E+00   -1.00000    1.00000 1QUAT, GROUP CYL
  2  0.0000000E+00  0.0000000E+00   -1.00000    1.00000 2QUAT, GROUP CYL
  3  0.0000000E+00  0.0000000E+00   -1.00000    1.00000 3QUAT, GROUP CYL

DF#  SCALED VALUE  UNSCALED VALUE DESCRIPTION
---  ------------  -------------- -----------
  1   10465.17       10465.17     XX MOMENT OF INERTIA

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> RUN GRG 5                        run 5 GRG iterations
OBJECTIVE FUNCTION GRADIENT
-0.35527E-12  7701.1      7705.8
LAGRANGIAN FUNCTION GRADIENT
 0.00000E+00 0.00000E+00 0.00000E+00
LENGTH OF SEARCH DIRECTION VECTOR=    1.41421
SEARCH DIRECTION
 0.48916E-16-0.70711    -0.70711
LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 4.0000E-02  searching for the
LINE SEARCH STEP  3 STEP LENGTH= 8.0000E-02  optimum
LINE SEARCH STEP  4 STEP LENGTH= 1.6000E-01
LINE SEARCH STEP  5 STEP LENGTH= 3.2000E-01
LINE SEARCH STEP  6 STEP LENGTH= 5.7000E-01
```

```
FITTING PARABOLA
LINE SEARCH STEP  7 STEP LENGTH= 2.7876E-01

DESIGN NUMBER      1
   SCALED STEP LENGTH              =  0.2787571
   SCALED OBJECTIVE      (DF#  1) =   10459.79
   ANALYSIS CALLS =      7     GRADIENT CALLS =      1
UPDATING HESSIAN
OBJECTIVE FUNCTION GRADIENT
 0.49921       7135.7      7140.6
LAGRANGIAN FUNCTION GRADIENT
 0.49921       7135.7      7140.6
LENGTH OF SEARCH DIRECTION VECTOR=    1.13546
SEARCH DIRECTION
-0.43432E-03-0.70711    -0.70711
LINE SEARCH STEP  1 STEP LENGTH= 2.0000E-02
LINE SEARCH STEP  2 STEP LENGTH= 9.9985E-03

DESIGN NUMBER      2
   SCALED STEP LENGTH              =  0.0000000E+00
   SCALED OBJECTIVE      (DF#  1) =   10459.79
   ANALYSIS CALLS =     10     GRADIENT CALLS =      2

 OPTIMUM ASSUMED (MINIMUM STEPSIZE REACHED)  optimum reached.

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> PRO POS                   draw the optimal solution.
                                   the cylinder has indeed rotated
                                   so that the axis of symmetry lines
                                   up parallel with the global x axis
```

DESIGN MENU:PRO,DIS,SET,RUN,PAR,HIS,EXP,TOL,APP,COM,QUI,HELP
<SELECT> QUIT



Trouble Shooting Guide For NAVGRAPH Model Generation

PROBLEM:

Points appear to be coincident and cannot be selected accurately with thumb wheels or mouse select.

POSSIBLE SOLUTIONS:

-rotate and/or magnify until unique point can be selected.

PROBLEM:

Point Labels cannot be seen or read.

POSSIBLE SOLUTIONS:

-check to see if point labels are turned on (DISPLAY OPTIONS)
-turn point entities on
-rotate and/or magnify (first turn AUTO-CENT off)
-if the NAVGRAPH data base you are working with was translated from an AFATL/ASE format data base turn node entities on, node labels on, and HIDDEN on (under DISPLAY OPTIONS). Node numbers and node coordinates are the same as the point numbers and point coordinates
-plot selected points by using the PLOT POINT command

PROBLEM:

A line cannot be defined.

POSSIBLE SOLUTIONS:

-check to see if each point (not node) defining the line is defined in the data base by using the LIST POINT command
-be sure the correct number of points are given for the line
-no more than 30 points may be used to define a spline

PROBLEM:

A line or spline does not follow the shape of the surface or edge closely enough.

POSSIBLE SOLUTIONS:

-redefine the line by overwriting
-use a different type of line
-try using fewer, more, or different points
-define additional points on the surface or edge
-consider a different location for the edge
-consider using degenerate edges

PROBLEM:

A valid solid cannot be defined

POSSIBLE SOLUTIONS:

-make sure that the lines defining the solid share eight (8)

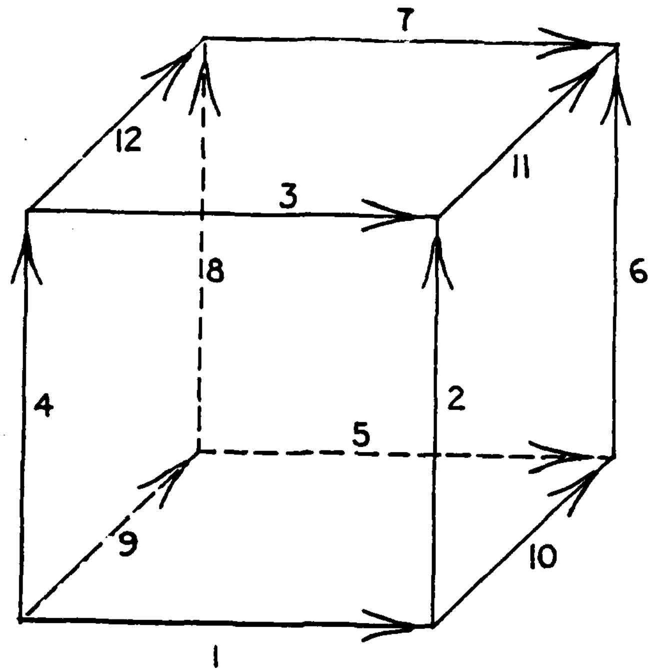


Figure 18: The edge direction and ordering convention for NAVGRAPH solids.

```
  common end points (at the corners)
 -make sure that the solid is composed of twelve (12) lines
 -assure that line directions are as shown in Figure 17
 -define surfaces one at a time to identify problem points or
  lines

PROBLEM:
     A draw command does not draw desired entities
POSSIBLE SOLUTIONS:
     -list entities to make sure they are in the data base
     -turn appropriate entities on
     -try drawing points only first

PROBLEM:
     A plot command does not respond.
POSSIBLE SOLUTIONS:
     -check to make sure desired entities are present in data base
     -enter a draw command first followed by a plot command
```

Suggestions and Tips

1. Before creating the model, make a rough sketch on paper of each line including direction, line, and point labels.
2. To save time, take advantage of EXTRUDE and SWEEP commands.
3. Creating half of a model that is symmetric around at least one axis is advisable. The model can then be reflected using the MIRROR command. This saves time and is usually more accurate.
4. Merging two or more lines can be helpful in curve fitting.
5. Use the surface fill option (under MORE OPTIONS) to see if a solid follows the shape of the desired object.
6. Become as familiar as possible with the shape of the object to be modelled in order to choose the best points for line and spline definition.
7. Try to define lines using points that are equally spaced.
8. 3-D graphics can be deceiving at time -- rotate and/or magnify to verify the correctness of the geometry. (Listing the data base may not always be helpful.)
9. To get a better idea of the shape of the object to be modelled, turn HIDDEN on and node and element labels off. View from several different angles.
10. View the shape as a deformed cube of several deformed cubes and try to place twelve (12) lines in a natural way.
11. Time can be saved by turning HIDDEN off when executing a draw command (the hidden line algorithm is by-passed).

Curve Fitting Technique

Figure 18 shows eight (8) points around the nose section of the F-15. Points 1, 3, 4, 5, 6, 9, 8, and 7 are at the surface and when joined by straight lines show the general shape. The other number labels on the figure label the finite elements. A spline is best suited to fit the shape. At the appropriate prompt 11 point numbers are given in this order: 7, 1, 3, 4, 5, 6, 9, 8, 7, 1, and 3. This method of overlapping beginning and ending points is necessary to give a smooth curve through point 1 (see Figure 19). This procedure creates 11 lines as follows: line 1 is the entire spline (it is defined as a merged line in the data base), lines 2-11 are the individual parametric cubic definition lines created between each pair of points. Lines 1, 2, and 11 can then be deleted to leave only one line between each pair of points (see Figure 20). These lines are now available for merging and creating solids.

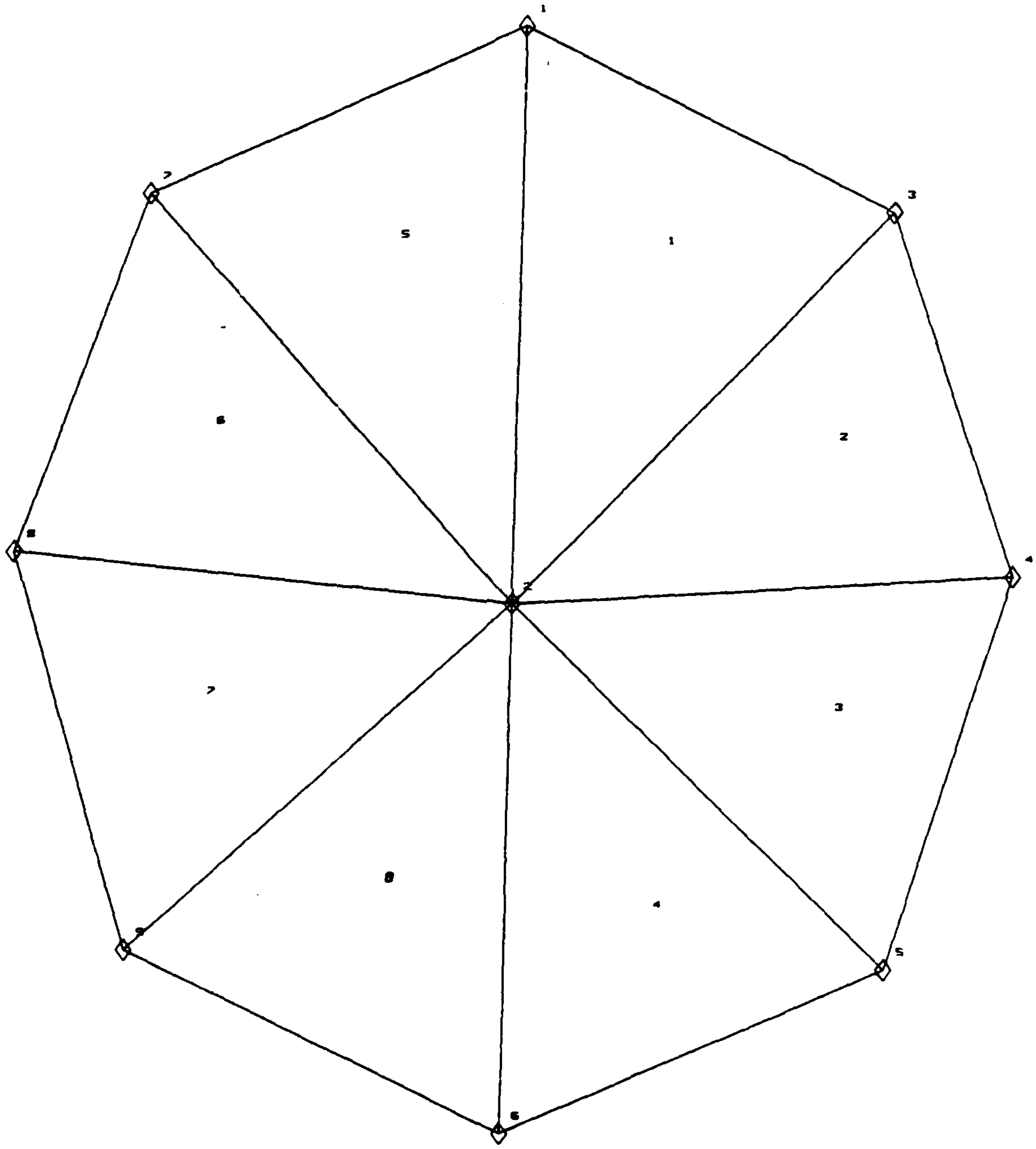


Figure 18.

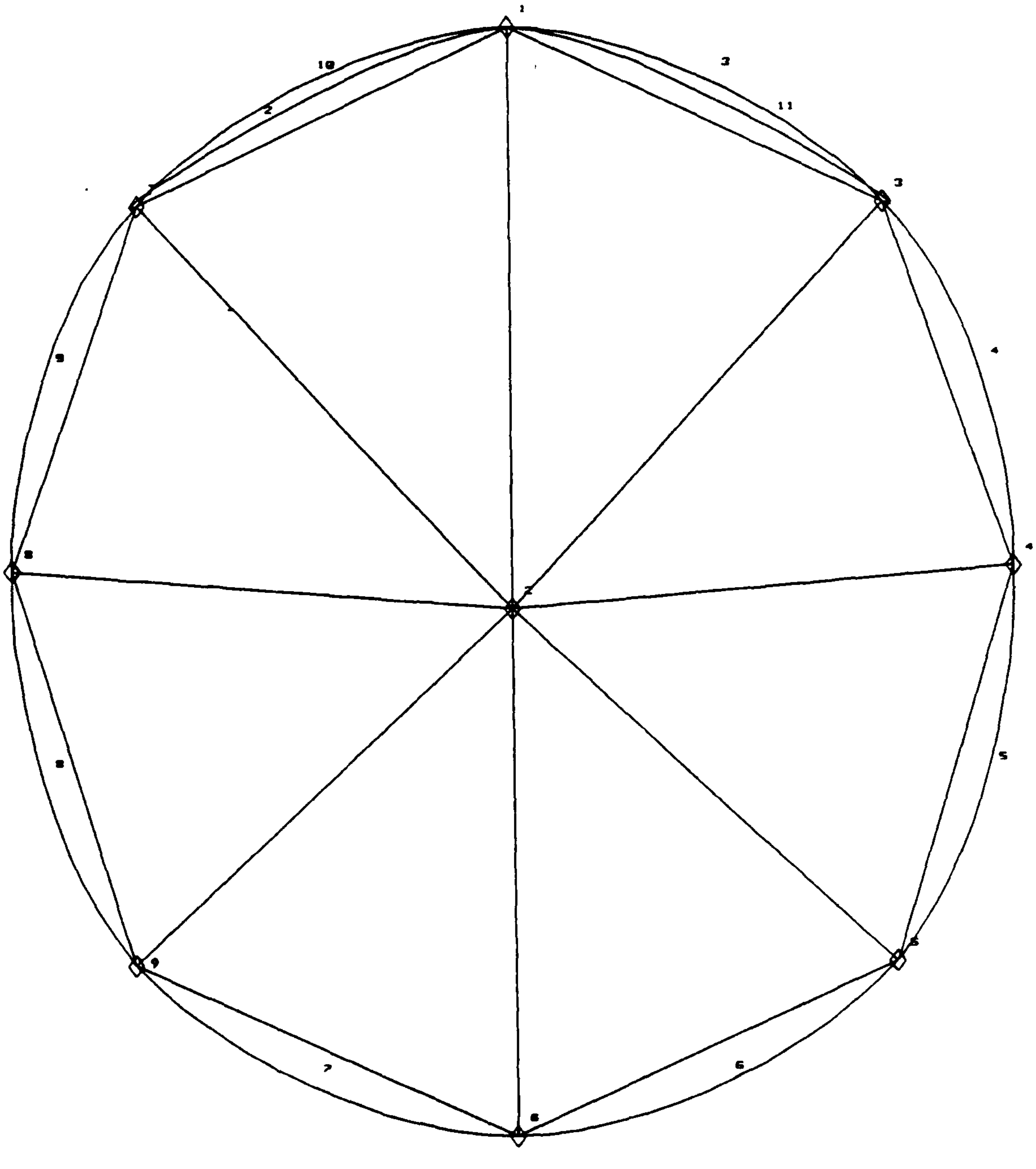


Figure 19.

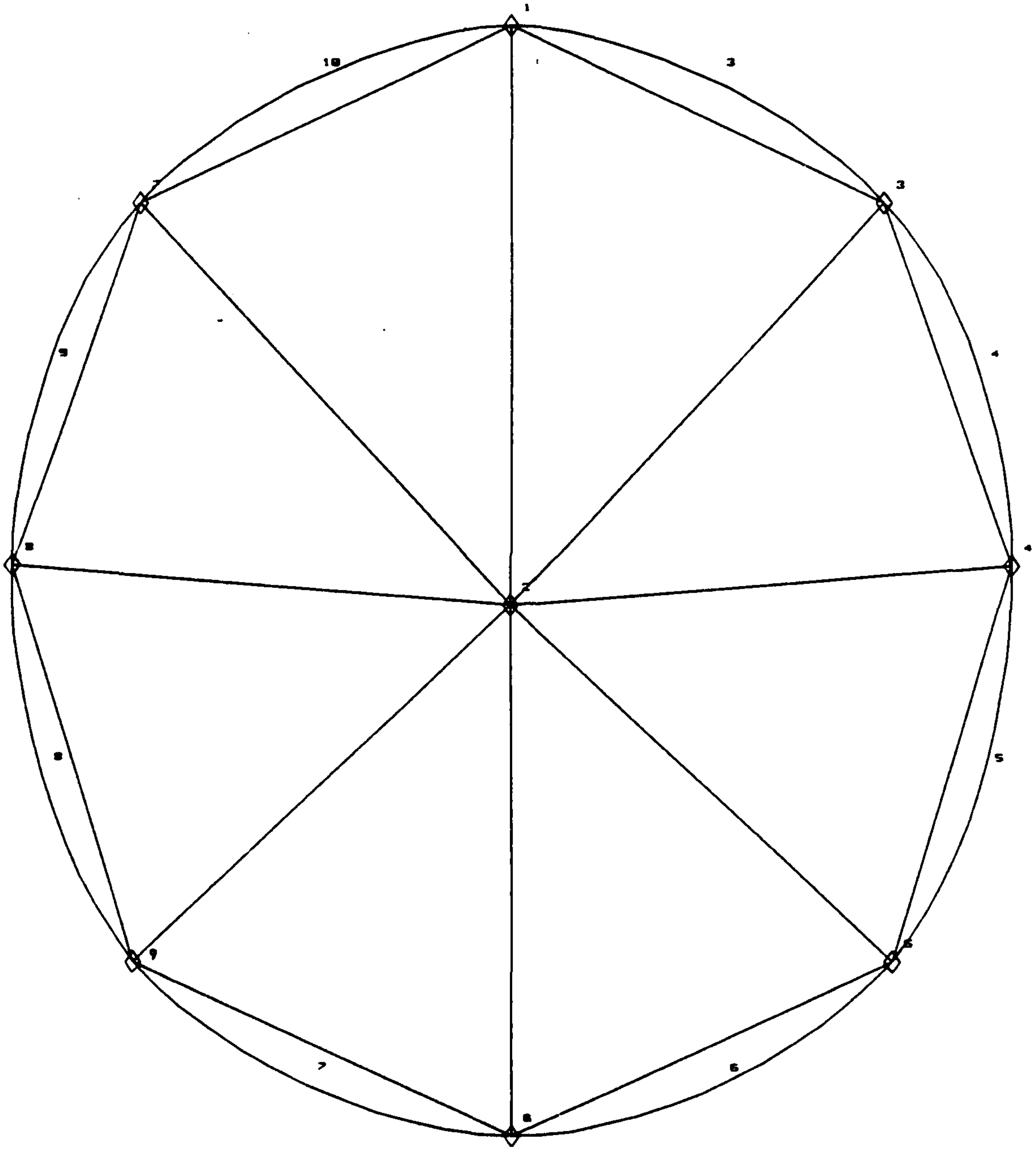


Figure 20.